(1)

Study
Note
94-02

The Army New Personnel System Evaluation Model

Jeffery L. Kennington, Farin Mohammadi, and Riad A. Mohammad

United States Army Research Institute
for the Behavioral and Social Sciences

October 1993

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Director

Research accomplished under contract for the Department of the Army

Jeffery L. Kennington
Farin Mohammadi
Riad A. Mohammed

Technical review by

Abraham Nelson

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 1993, October	3. REPORT TYPE AND DATES COVERED Final Sep 92 - Mar 93

4. TITLE AND SUBTITLE The Army New Personnel System Evaluation Model	5. FUNDING NUMBERS DAAL03-91-C-0034 65803D 730 2104 C15 D.O. 0542 TCN 92491
6. AUTHOR(S) Kennington, Jeffery L.; Mohammadi, Farin; and Mohammed, Riad A.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Private Consultants (No Organization) 6511 Wickerwood Dr. Dallas, TX 75248	8. PERFORMING ORGANIZATION REPORT NUMBER --
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Institute for the Behavioral and Social Sciences ATTN: PERI-RG 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARI Study Note 94-02

11. SUPPLEMENTARY NOTES

--

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE --

13. ABSTRACT *(Maximum 200 words)*

Manpower planning models have been used extensively by the Department of Defense for the past forty years. A manpower problem can be modeled as a Markov model, a linear goal program, or a combination of the two. Markov models use historical data to derive recruitment policies or to estimate the future structure of a manpower system. Linear goal programming models are used to evaluate the impact of changing a manpower policy, to determine recruitment policy, to forecast future budget requirements, and to establish the ability to significantly increase or decrease man strength in a short period of time. A combination of the two approaches is used to obtain optimal policies for a manpower system considering the cost and conflicting objectives. In this report, some of the models developed and techniques used for manpower planning are reviewed, and a new prototype model is developed. The model generator has been implemented in FORTRAN.

14. SUBJECT TERMS Manpower planning Linear programming Markov model Goal programming	15. NUMBER OF PAGES 60
	16. PRICE CODE --

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited

NSN 7540-01-280-5500 Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z39-18
298-102

THE ARMY NEW PERSONNEL SYSTEM EVALUATION MODEL

CONTENTS

CONTENTS (Continued)

THE ARMY NEW PERSONNEL SYSTEM EVALUATION MODEL

Introduction

As a result of the changes in the funding levels of the Department of Defense, the Army personnel system is currently undergoing significant change. The overseas manpower requirement will be reduced and the rotation policies will need to be evaluated. The permanent change of station (PCS) policies need to be reexamined.

This report gives a survey of techniques that have been used in personnel policy analysis along with a prototype mathematical model. The prototype model demonstrates the key features needed in a production model, along with an illustration of how such a production model could be developed. We also give the model generator code.

The Survey

Manpower planning is concerned with the allocation of the right number of personnel to different tasks in order to achieve short and long term goals of an organization without violating organizational policies. The extensive use of the term manpower planning started after World War II when the U.S. Navy began reorganization of its technical and managerial manpower. Manpower planning models have been used for government and public agencies and extensively in the military due to the importance of national defense planning and budget issues. Accurate forecasts and eminent knowledge of market trends are necessary for a successful development of a manpower system.

Markovian models have been used to estimate grade-wise distribution of future manpower as well as to maintain new recruitment and promotion or firing policies (see Vajda [1976], Abodunde and McClean [1980], Zanakis and Maret [1980], Bartholomew [1982], Edwards [1983], and Raghavendra [1991]). A combination of Markov models and linear programming is used to obtain optimal policies taking into account not only the manpower requirements but also costs and conflicting objectives (see Young and Abodunde [1979], and Zanakis and Maret [1981]). Dynamic programming techniques and optimal control theory may also be used in manpower planning but are not commonly used (see Edwards [1983]).

Vajda [1976] and Edwards [1983] model personnel movement within an organization as a Markov chain. Vassiliou [1976] develops a Markov model to analyze wastage, departures, from an organization. Young and Abodunde [1979] represent internal and external transitions of personnel within a graded organization as a discrete time deterministic model and present a linear programming formulation of the problem to obtain optimal long term recruitment policy. Abodunde and McClean [1980] analyze the effects of possion recruitment on a stochastic version of the Young and Abodunde model. Zanakis and Maret [1981] discuss different stages of development and validation of a Markov model, and combine a Markov model and linear goal programming for manpower planning. Edwards [1983] conducts a survey of manpower planning applications in the U.K. and other European countries. The survey includes different software available for manpower planning, the models and the techniques used in their development as well as requirements for using this software. Hall [1986] proposed an aggregate manpower planning system using graphical models. Smith and Bartholomew [1988] present a historical review that traces the growth of manpower planning in the U.K. since World War II. Raghavendra [1991] uses a Markov chain model to estimate the probability matrix for promotions in an organization. He then derives the formulas for obtaining a minimum level of seniority and performance rate required for promotion.

The needs for manipulating a multi-characteristic pool of personnel over a long planning horizon has led to a number of linear programming based approaches for evaluating the alternative solutions. The linear goal programming model (LGPM) provided an indispensable aid for managing the flow of personnel in a manner that best meets the desired manning structure and quality over the planning years.

Several LGPM based pragmatic systems have been developed and extensively exploited in the US Army and Navy. Two such systems are the Accession Supply Costing and Requirement (ASCAR) and Enlisted Loss Inventory Model-Computation of Manpower Programs Using Linear Programming (ELIM-COMPLIP) models (see Collins, Gass and Rosendahl [1983], and Holz and Worth [1980]). The primary use of these systems is to evaluate the impact of changing the manpower policy, such as variations in the promotion and separation rates and desired force compositions, on the overall performance and man-strength. Also, they have been used to provide the capability to determine the number of new recruits with certain qualifications to meet future needs, to improve the ability to forecast the budgets required to cover certain future manning levels, and to establish the ability to significantly increase or reduce the man-strength in a short period of time to meet critical situations such as Vietnam and the Gulf war.

Goal priorities are very decisive in affecting the solution profile. These priorities can be incorporated in the model by associating suitable weights with the goal deviation variables. Determining these weights directly is not feasible, Gass [1986] developed an automated system for calculating composite weights, that properly reflect priorities of composite goals, using Saaty's analytical hierarchy process (AHP).

In the following section we present a summary of some of the Markovian approaches to the manpower planning problem, followed by a summary of linear goal programming approaches. A combined approach is presented in the last section.
Markov Approaches

Markov Approaches

Manpower planning consists of demand prediction, supply prediction, and development of policies to reconcile any difference between supply and demand. Future demands can be predicted by simple extrapolation, regression, factor analysis, work study, or other forecasting techniques. To predict the future supply one may represent the manpower system as stocks and flows, where stocks are number of employees in each category and flows are movement of personnel into, within, or out of the system. While most flows are under complete managerial control, voluntary leaves cannot be totally controlled by management. An indicator of the voluntary leave survival function may be obtained using cohort or census analysis. If the future demand has been predicted and stocks and flows determined, possible shortage or excess can easily be obtained (see Edwards, 1983).

A mathematical statement of the problem is made possible by classifying the personnel into different groups by their sex, race, level of employment, education, etc., and then representing these groups as different states of a Markov model. The transition probability matrix (TPM) represents the movements within the organization such as promotions and transfers or any movement of the personnel from, to, or within an organization. States such as retirement, voluntary and involuntary wastage are absorbing states, while others are transient states. To model a manpower system as a markov chain the following steps must be followed.

(1) Divide the planning horizon to small intervals. Small intervals may lead to more accurate estimates, but the TPM of a very short time interval can prevent the stationarity of the TPM. Zanakis and Maret [1980] suggest that the stage interval should be determined based on objectives of the study and the planning horizon. For long-range plans usually a yearly time interval is selected.

(2) Define states of the system. The states of a Markov chain must be mutually exclusive and exhaustive, ie., states should include each possible case once and only once. For example, states of a Markov chain can be defined as levels of employment within an organization. These states are exhaustive and mutually exclusive since each employee has a rank and only one rank.

(3) Collect data on the number of transitions from each state to other states to estimate the TPM. In the absence of the transition data least square estimates of TPM may be obtained using the historical data on the number of personnel at each state at different periods.

(4) Estimate the TPM and validate the model. The sample size determines the accuracy of the TPM estimates. χ^2 tests are used to validate the model.

A stochastic model is presented by Vassiliou [1976] for wastage analysis in hierarchically structured manpower systems. In his paper Vassiliou classifies the reasons for leaving an organization as (1) retirement or death, (2) discharge, and (3) resignation. He assumes a constant transition probability for discharge, retirement, or death over the time period. He then tests his assumption and finds no evidence to reject it for the cases he studies. The expected number of voluntary leaves he is equal to the expected number of "normal" departures, increased by the number of "frustrated" potential departures proportional to the cumulative rate of contraction of the organization and decreased by the number of "not frustrated" potential departures proportional to the cumulative rate of expansion of the organization. He develops necessary equations for this model and examines the model using the data from two different firms. Then he compares his results with results obtained from a deterministic model and concludes that his model describes the wastage flow properly and produces better results.

Young and Abodunde [1979] investigate the consequences of a controlled hiring policy over a long period of time for a discrete time model assuming that the demand level reaches a steady state after a sufficiently long time. By assigning costs to over- and under- production and the use of mathematical programming techniques, they develop a linear programming model to obtain an optimal recruitment policy.

Assuming a constant promotion and wastage level, Abodunde and McClean [1980] study the effects of recruitment that follows a Poisson process. They provide expressions to obtain the number of personnel at each grade level at each time period given the desired steady state size of a telephone system.

Raghavendra [1991] uses a Markov chain model to obtain promotion policies under certain assumptions. He then translates promotion policies into seniority levels and performance rates required for promotion.

Markov Chain Models

The use of a Markov chain in modeling military, government, and public agencies manpower supply goes back as far as late 1950. In this section we present a Markov chain model presented by Raghavendra [1991]. This model is developed under the assumption that no double promotion or demotion is permitted and that at the beginning of each period a constant proportion of staff are to be promoted to each level.

Let t represent the planning periods, T the planning horizon, i and j the states, and K the number of states in the system, $N_j(t)$ the number of staff in grade j at the beginning of the period t, $P_{ij}(t)$ the probability that a member of staff in grade i at the beginning of period t will be in grade j at the beginning of the next period, $R_j(t)$ the number of new recruits to grade j during period t, $W_j(t)$ the wastage factor representing the proportion of members of staff in grade j leaving the system during period t due to retirement, death, resignation, etc., and e_j the ratio of staff promoted to state j to the total staff who have joined state j (promoted and recruited). Then under the above assumptions the following equations hold:

$$N_j(t+1)=\sum_{i=1}^{K} P_{ij}(t)N_i(t)+R_j(t); \quad \forall\ j=1,2,\ldots,K. \tag{1}$$

$$\sum_{j=1}^{K} P_{ij}(t)+W_i(t)=1; \quad \forall\ i=1,2,\ldots,K. \tag{2}$$

$$P_{ij}(t)=0; \quad \forall\ j>i+1\ ,j\le i-1. \tag{3}$$

Given the current personnel structure $N(1)$, desired future structures $N(t)$, and wastage factors $W(t)$, for all periods under study, and given the proportion of staff to be promoted to each state, e_j, the promotion and recruitment policies for K, then the higher level of employment possible can be developed using the following equations.

$$N_K(2)=P_{(K-1)K}(1)N_{K-1}(1)+P_{KK}(1)N_K(1)+R_K(1). \tag{4}$$

From (2) and (3)

$$P_{KK}(1)=1-W_K(1), \tag{5}$$

therefore

$$P_{(K-1)K}(1)N_{K-1}(1)+R_K(1)=N_K(2)-N_K(1)(1-W_K(1)) \\ =\hat{N}_K(2) \tag{6}$$

Since the proportion of staff promoted to state K is e_k and the proportion of staff to be recruited to grade K is $1-e_K$ it follows that

$$P_{(k-1)K}(1)N_{K-1}(1)=e_K\hat{N}_K(2), \tag{7}$$

and

$$R_K(1)=(1-e_K)\hat{N}_K(2). \tag{8}$$

From (7)

$$P_{(K-1)(K-1)}(1)=\left[\frac{e_K\hat{N}_K(2)}{N_{(K-1)}(1)}\right], \tag{9}$$

and

$$P_{(K-1)(K-1)}(1)=1-W_{(K-1)}(1)-P_{(K-1)K}(1). \tag{10}$$

Once estimates of transition probabilities and number of people to be hired to state K at the beginning of time period 2 are obtained, similar information for other time periods can easily be obtained. For a numerical example see Raghavendra [1991].

Model Validation

The flow of personnel is modeled as a Markov chain by first, selecting the stage interval, second, defining the states, third, collecting data, and forth, estimating the TPM. Once the model is developed it must be validated. The TPM is estimated by converting the number of transitions to row proportions. This calculation yields a maximum likelihood estimate of the true TPM if the process is stationary, ie., constant over time. Zanakis and Maret [1980] present the following $\chi 2$ tests to investigate the stationarity of the TPM and its elements. Let s be the number of nonabsorbing states, m be the total number of absorbing and nonabsorbing states, T the be number of stages observed, and $n_{ij}(t)$ be the number of transitions from state i to state j during period t. Then $n_i(t)$, the total number of people available in state i at the beginning of period t, $\hat{p}_{ij}(t)$, the ratio of people promoted to state j from state i in time period t, and $\hat{p}_{ij}$ the hypothesized stationary probability of an (i,j) transition can be described as follows:

$$n_i(t) = \sum_{j=1}^{m} n_{ij}(t) \quad (11)$$

$$\hat{p}_{ij}(t) = n_{ij}(t)/n_i(t) \quad (12)$$

$$\hat{p}_{ij} = \sum_{t=1}^{T} n_{ij}(t) / \sum_{t=1}^{T} \sum_{j=1}^{m} n_{ij}(t) \quad (13)$$

The following χ^2 tests may be used to test the stationarity of the TPM and its elements at the α level of significance.

(1) The (i,j) transition probability is constant over time if

$$\sum_{t=1}^{T} n_i(t)[\hat{p}_{ij} - \hat{p}_{ij}(t)]^2/\hat{p}_{ij} < \chi_\alpha^2[T-1]. \quad (14)$$

(2) Transitions to a column state j are stationary if

$$\sum_{i=1}^{s} \sum_{t=1}^{T} n_i(t)[\hat{p}_{ij} - \hat{p}_{ij}(t)]^2/\hat{p}_{ij} < \chi_\alpha^2[s(T-1)]. \quad (15)$$

(3) Transitions from a row state are stationary if

$$\sum_{j=1}^{m} \sum_{t=1}^{T} n_i(t)[\hat{p}_{ij} - \hat{p}_{ij}(t)]^2/\hat{p}_{ij} < \chi_\alpha^2[(m-1)(T-1)]. \quad (16)$$

(4) The entire TPM is constant over time if

$$\sum_{i=1}^{s} \sum_{j=1}^{m} \sum_{t=1}^{T} n_i(t)[\hat{p}_{ij} - \hat{p}_{ij}(t)]^2/\hat{p}_{ij} < \chi_\alpha^2[s(m-1)(T-1)]. \quad (17)$$

Markov models are used to obtain policies to achieve a specific goal (structure of personnel) with no consideration for the cost, possible constraints, or conflicting objectives. In the following section the goal programming approach along with three models developed for US Army and the US Navy are presented.

Linear Goal Programming

Manpower planning can be formulated as a linear goal-programming model, where each goal to be attained is represented by one constraint. The model objective is to obtain a compromising solution that will satisfy all the goals as close as possible, when the exact attainment of all the goals simultaneously is impossible.

Let x_j denote the value of the j^{th} decision variable. Let G_i denote the i^{th} goal. Let I denote the number of goals to be achieved, and J denote the number of decision variables. Let a_{ij} denote the per unit contribution of the j^{th} decision variable into goal i. Let g_i^- denote the goal under-achievement variables, and g_i^+ denote the goal over-achievement variables; g_i^- and g_i^+ are called deviation variables. Let w_i^- and w_i^+ represent the priority weights of under- and over-achievement of goal i. Therefore, a general formulation of the goal-programming model can be represented as follows:

$$\textit{Minimize} \quad \sum_{i=1}^{I} w_i^+ g_i^+ + \sum_{i=1}^{I} w_i^- g_i^- \tag{18}$$

$$s.t. \quad \sum_{j=1}^{J} a_{ij} x_j + g_i^- - g_i^+ = G_i; \qquad i=1,\ldots,m, \tag{19}$$

$$x_j, g_i^+, g_i^- \geq 0. \tag{20}$$

The determination of these priority weights is presented in the next section.

Many manpower planning problems can be formulated as large goal programs with hundreds of constraints representing the targets and limitations. Several practical models have been developed for these problems, such as the ASCAR and the ELIM-COMPLIP models. These models and a Navy model are presented in the following sections.

A US Navy Model

The manpower requirement for the US Navy is embodied in the present and future needs for a wide mix of specialties, with different years of commissioned service. The US Navy has several commissioning sources that have different capacities and costs. Some sources, such as the US Navy Academy (USNA), provide officers with wide specialty areas, while others produce officers with a single specialty. Therefore, to meet the future needs of experienced officers with different specialties, a proper mix of the commissioning sources must be established. A flexible approach is needed to provide a fast response to major changes in manpower policies, and to study different scenarios of source mix and inventory costs.

<u>The Approach.</u> In this model, officer inventory within each community is specified by the number needed in different states at each time period. An officer state is determined by three factors: (1) warfare community, (2) commissioning program, and (3) years of commissioned service (YCS). Since most communities follow a common promotion path in promoting their officers at known YCS experience levels, YCS is then used as a substitute for grades.

The expected flows between states in successive time periods is projected, in a markovian trend, using transition rates estimated from historical data. New accessions are also added to these flows.

<u>Formulation.</u> The model developed by Bres, Burns, Charnes, and Cooper [1980] utilizes discrete time periods. The subscripts used in this model are defined as follows:

i = Warfare community,
j = Commissioning source,
t = Time period,
k = Number of years of commissioned service, and
m = level of experience which is defined by lower and upper limits on YCS.

The constants used in this model are defined as follows:

I = Number of communities,
J = Number of commissioning sources,
K = Maximum length of service,
M = Number of experience levels,
T = The planning time horizon,
$I^0_{ij}(k)$ = Initial officer inventory in community i, form source j, with k YCS,
$S^t_{ij}(k)$ = Officer survival rate for t time periods in community i, from source j, with k YCS,
$G_{im}(t)$ = Officers strength goal in community i, for experience level m, in time period t,
$U(t)$ = Upper limit on total officer inventory in period t,
$B(t)$ = Budget limit for pay and allowances in period t,
$c_{ijk}(t)$ = Cost for pay and allowances for an officer in community i, from source j, with k YCS, in time period t,

l_{im} = Lower limit for YCS in experience level *m* for community *i*,
u_{im} = Upper limit for YCS in experience level *m* for community *i*,
$X(t)$ = Upper limit on the total number of officers that may be commissioned in time period *t*,
$P_j(t)$ = Maximum number of officers that can be commissioned from source *j* in time period *t*,
$Q_j(t)$ = Minimum number of officers that can be commissioned from source *j* in time period *t*,
$R_i(t)$ = Maximum number of newly commissioned officers that can be trained in community *i*, in time period *t*,
$P_{ij}(t)$ = Maximum allowable number of officers commissioned from source *j* to community *i*, for time period *t*,
$Q_{ij}(t)$ = Minimum allowable number of officers commissioned from source *j* to community *i*, for time period *t*,
$w^+_{im}(t)$ = Weight given to over-achievement of officer strength goal for community *i*, experience level *m*, time period *t*,
$w^-_{im}(t)$ = Weight given to under-achievement of officer strength goal for community *i*, experience level *m*, time period *t*, and
$w^-(t)$ = Weight given to negative deviation from total officer strength limit, time period *t*.

The decision variables used in this model are defined as follows:

$y_{ijk}(t)$ = Officers inventory in community *i*, from source *j*, with *k* YCS, at the beginning of time period *t*.
$x_{ij}(t)$ = Accessions to community *i* from source *j* in time period *t*.
$g^-_{im}(t)$ = Goal under-achievement for community *i*, experience level *m*, in time period *t*.
$g^+_{im}(t)$ = Goal over-achievement for community *i*, experience level *m*, in time period *t*.
$g^-(t)$ = Negative deviation from the total officer inventory upper limit in period *t*.

The constraints that define the relation between officer inventories and goals, for each community at different experience levels, are expressed as:

$$\sum_{j=1}^{J}\sum_{k=l_{im}}^{u_{im}} y_{ijk}(t) + g^-_{im}(t) - g^+_{im}(t) = G_{im}(t)\,, \tag{21}$$

where inventories can be expressed in terms of beginning inventories as:

$$y_{ijk}(t) = S^t_{ij}(k-t)\, I^0_{ij}(k-t) \qquad for\ \ 2 \le t \le k, \tag{22}$$

and in terms of subsequent accessions as:
The limitation on total officer inventories is expressed by:

$$y_{ijk}(t) = s_{ij}^{k}(0)\, x_{ij}(t-k) \qquad \textit{for} \quad t>k. \tag{23}$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} y_{ijk}(t) + g_{im}^{-}(t) = U(t) \quad \textit{forall } t. \tag{24}$$

The budget limits are expressed by:

$$\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} c_{ijk}(t) y_{ijk}(t) \le B(t) \quad \forall\ t. \tag{25}$$

Restrictions on the officer accessions from various warfare communities are addressed in several constraints. The limitation on the total number of the officer accessions in each period is expressed by:

$$\sum_{i=1}^{I}\sum_{j=1}^{J} x_{ij}(t) \le P(t) \quad \forall\ t. \tag{26}$$

In each community, the training capacity limits for newly commissioned officers are expressed by:

$$\sum_{j=1}^{J} x_{ij}(t) \le R_i(t) \quad \forall\ i,t. \tag{27}$$

The operating upper and lower limits for the commissioning sources are expressed by:

$$Q_j(t) \le \sum_{i=1}^{I} x_{ij}(t) \le P_j(t) \quad \forall\ i,t. \tag{28}$$

The upper and lower limits on the distribution of newly commissioned officers from each source to each community are expressed by:

$$Q_{ij}(t) \le x_{ij}(t) \le P_{ij}(t) \quad \forall\ t. \tag{29}$$

Let $\alpha(t)$, $\beta(t)$, $\gamma(t)$, and $\delta(t)$ denote the minimal and maximal proportions of change allowed between adjacent time periods. The proportional changes allowed for inputs to each community are expressed as follows:
for the accessions to community i. The proportional changes allowed for outputs from each source are expressed as follows:
for the output of source j. Finally, a nonnegativity condition is imposed.

$$\alpha(t)\sum_{j=1}^{J} x_{ij}(t) \le \sum_{j=1}^{J} x_{ij}(t+1) \le \beta(t)\sum_{j=1}^{J} x_{ij}(t) \quad \forall\ t \text{ and} \qquad (30)$$

$$\gamma(t)\sum_{i=1}^{I} x_{ij}(t) \le \sum_{i=1}^{I} x_{ij}(t+1) \le \delta(t)\sum_{i=1}^{I} x_{ij}(t) \quad \forall\ t \text{ and} \qquad (31)$$

The model objective is to find an officer accession and an arrangement plan that minimizes weighted deviations from officer strength goals. This can be expressed as follows:

$$\textit{Minimize} \quad \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{m=1}^{M} (w_{im}^{+}(t)g_{im}^{+}(t)+w_{im}^{-}(t)g_{im}^{-}(t)) + \sum_{t=1}^{T} w^{-}(t)g^{-}(t), \qquad (32)$$

subject to constraints (21)-(31).

As an illustration, the model was applied to four communities (Unrestricted Line officers) for the first ten years of commissioned service omitting the budget constraint. It provided a satisfactory accession plan with minimal deviations for the high priority requirements. For more details see Bres, Burns, Charnes and Cooper [1980]. The model can also be used to provide the impact on budgets if the appropriate results are substituted into constraint (25) to study funds-flow consequences of these plans.

The ASCAR Model

The Accession Supply Costing and Requirement (ASCAR) model is a goal programming based model. It was initially developed by General Electric for the Congressional Budget office in the late seventies. The model was then revised and used by the office of the Assistant Secretary of Defense. The main objective of the ASCAR project was to provide the capability to investigate the effects of changing manpower policies on personnel requirements, military qualifications, and the associated costs.

The ASCAR model consists of data management programs, a data base, personnel flow simulation routines, a cost routine, and a report routine. It follows a five-step process to evaluate the annual accessions necessary to meet the strength and quality requirements, and to calculate the associated costs. These steps are summarized in the following:

- Initially, the historical data are analyzed to develop flow rates and starting personnel levels for the simulated period.

• These rates and levels are then used in the personnel flow simulation routines to compute losses, to forecast the supply of new recruits for each simulated year, and to simulate the flow of new recruits in order to calculate the required accessions.

• The supply of new recruits is categorized by sixty qualitative factors such as educational level and mental category.

• The goal programming procedure optimizes the different category mix of new recruits to match the required man-strength and characteristics while satisfying the specified constraints.

• The cost module produces the cost estimates for the analyzed policy using the cost factors and the projected strength determined in the previous steps.

The process is then repeated for each additional year. More details are provided in Collins, Gass, and Rosendahl [1983].

<u>The ELIM-COMPLIP Model</u>

The Enlisted Loss Inventory Model-Computation of Manpower Programs Using Linear Programming (ELIM-COMPLIP) is a goal programming based forecasting system developed by General Research Corporation for the Department of the Army. It is used for manpower planning, budgeting, and personnel policy formulation. The system is described in details by Holz [1980].

In the late sixties the Assistant Secretary of the Army, William K. Brehm, was addressing several "what if" questions related to different manpower policies for Vietnam. Each alternative required approximately 8-20 computation hours. Therefore, the General Research Corporation was asked to automate these computations. One of the two systems developed was the COMPLIP optimization system. The main objective of COMPLIP is to minimize the weighted sum of the deviations of the actual man-strength from the required goals. It provided the fast responsiveness needed to generate accurate manning plans for models with conflicting constraints that were difficult to approximate by manual calculations. The second was a simulation of the manual system. It was never used after the demonstration of the COMPLIP system. COMPLIP provided upper and lower bounds on draft calls that were very useful in preventing fluctuations to this sensitive political issue.

In the fiscal year 1972, the Army was phasing down from Vietnam, when the Congress passed an authorization bill to reduce the Army's strength by 50,000 for that year, which significantly accelerated the phase-down plan. Several drastic policies were adopted that disrupted personnel management and caused the strength to fall below authorized level. The Army ended the year 50,000 men below the desired strength. As a response to this crisis, the General Research Corporation developed the Enlisted Loss Inventory Model (ELIM) using a new loss projection method. The new method starts with the most recent information and preserves the relations between end strength, gains, and losses to avoid the risks created with the previously used process.

The ELIM-COMPLIP system is self-correcting with respect to projection errors, and provides accurate strength forecasts. It includes five modules: data processor, rate generator, inventory projection, optimization, and a report generator module. The primary task of the rate generator is to exponentially smooth the time series of historical data to project loss rates. These rates are used in the inventory projection module to compute retention rates that develop the goal programming coefficients. For more details see the description and the system schematic in Holz [1980].

Goal Weight Evaluation

Linear goal-programming has proved to be very effective in solving multi-criteria problems, such as manpower planning problems. Usually, the objective function (18) of a general linear goal-programming model (18)-(20) contains several thousands of deviation variables. Most problems have specific properties that necessitate proper determination of weights, w_i^- and w_i^+, associated with the deviation variables to reflect the correct goal priority. In general, direct weight determination for small problems with related goals is not a hard task. However, direct weight determination for thousands of dissimilar goals is not viable, as it resembles comparing "oranges to apples". In this section we represent a weight determination approach proposed by Gass [1986] using Saaty's analytical hierarchy process (AHP).

Consider the grade-skill goal constraints (21), budget constraints (25), and promotion goal constraints (30). Meeting any of these constraints has a completely different meaning than meeting the other two. Also, meeting a grade-skill goal in one year may have a much different priority than the same goal in a different year. Therefore, planners must be able to designate the objective goal weights that cause a proper transition to be made over the planning horizon to reach specified requirements at certain years. Undoubtedly, the exact matching of man-strength requirements that fluctuate over the intended period is not achievable. However, a compromising solution can be accomplished where the goal weights will determine the years to be closely matched, namely, the solution profile.

Gass [1986] developed an automated routine for calculating thousands of weights necessary to determine an acceptable solution for the linear goal program (18)-(20).

A grade-skill model with a seven year planning horizon is used to illustrate Gass' method. In this model the goals are defined by (1) total personnel at the end of each year, (2) grade-skill qualifications, (3) promotion goals by grade-skill, and (4) loss goals by grade-skill. These goals can be represented by a hierarchical structure of factors illustrated in Figure(1). Level 1 represents the model main objective. Level 2 contains the major requirements (year end-strength) that influence the model. Level 3 contains the grade-skill, promotion, loss, and gain targets that influence level 2.

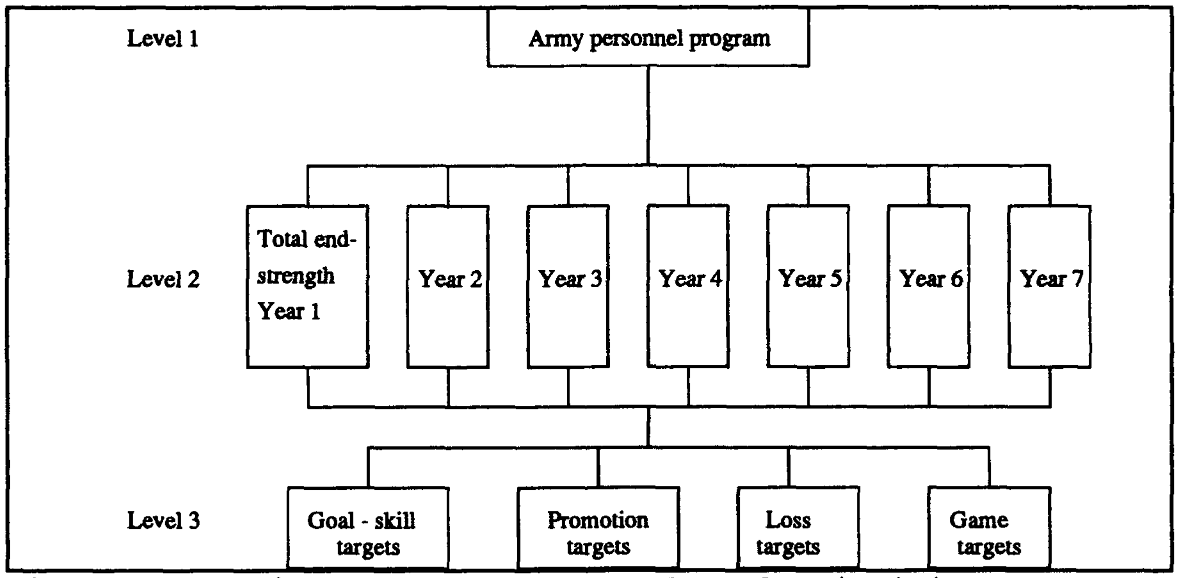


Figure 1. A hierarchy for personnel goal priorities (Gass, 1986)

The Gass method establishes the overall priorities in the main objective from the hierarchical factor priorities by systematically comparing the factors within each level. In each level, the weights that reflect the factor priorities are obtained from a comparison matrix associated with the level. The matrix elements have values 1 to 9 and their reciprocals, and represent the importance of meeting each factor. A typical comparison matrix corresponding to level 2 of the above example is:

$$A = \begin{bmatrix} 1 & 5 & 3 & 1 & 3 & 3 & 3 \\ 1/5 & 1 & 1/5 & 1/3 & 1/5 & 1/7 & 1/9 \\ 1/3 & 5 & 1 & 7 & 7 & 9 & 9 \\ 1 & 3 & 1/7 & 1 & 3 & 5 & 7 \\ 1/3 & 7 & 1/9 & 1/5 & 1/3 & 1 & 5 \\ 1/3 & 9 & 1/9 & 1/7 & 1/3 & 1/5 & 1 \end{bmatrix}$$

where $A_{1,4}=1$ means that year 1 and year 4 are equally important, and $A_{7,2}=9$ means that year 7 is significantly more important than year 2, with respect to the model main objective. The Gass method determines the solution to the system of equations $Aw= \lambda w$, which is the well-known eigenvalue problem. Following the AHP theory, for λ equal to the largest eigenvalue, the normalized w_i components are then interpreted as weights that represent the importance of each factor. For the above matrix we have the following weights for level 2:

Year	1	2	3	4	5	6	7
Weights	0.25	0.03	0.38	0.18	0.07	0.08	0.01

These weights indicate that the accomplishment of year 3's goal has the highest priority (0.38), then year 1's goal and so on.

A similar analysis is carried out to determine the level 3 factor priorities for each year. The composite goal priorities for the level 3 items are developed by multiplying the year's priorities by the corresponding factor priority, and then adding. A detailed example is described in Gass [1986].

Finally, the AHP priorities are converted to goal weights for each target-year combination. This is accomplished using closeness factors. Each target-year combination is characterized by an ordered couple $c(i,j)$, in this example $i=1,...,4$ (targets) and $j=1,...,7$ (years), which adjusts the composite weight for target i to account for the priority of year j. Thus, weight for any goal i and year j is determined by

$$w_{ij}=S_i^t-c(i,j)\left(\frac{S_i^t-S_i^b}{5}\right), \tag{33}$$

where S_i^b is the top of the scale for goal i and S_i^t is the bottom of the scale for target i. A broad scale (e.g. 0 to 1000) is used so that the optimization algorithm would be able to differentiate properly between the goals.

The AHP theory can be extended to determine the skill weights for each grade, in order to set priorities between the skills within a grade for each year. An example of weights indexed by grade and skill is given in Gass [1986].

Markovian Goal Programming

Even though Markov models are successfully used to estimate the future manpower plan or to develop policies to obtain a desired future personnel structure, they cannot consider cost, constraints, or conflicting objectives. Zanakis and Maret [1981] use a combination of the Markovian and LGP techniques to obtain a manpower plan. As previously stated they use historical data to estimate TPM and desired structure of the personnel system. Once the number of personnel is determined they develop a goal programming model to satisfy the organizational goals such as: keep the cost at the lowest possible level, hire a specific number of recent college graduates, keep the ratio of number of personnel at a specific level of employment to other levels constant, and other similar goals. For details and a numerical example see Zanakis and Maret [1981].

Assuming that costs can be assigned to under and over production and given the time demand level will reach steady state, Young and Abodunde [1979] present a formulation for a discrete-time Markovian goal programming model.

Let $G(t)$ denote the target demand at time period t, and $C^-(t)$ and $C^+(t)$ denote the costs ascribed to the goal under- and over-achievement. Let $R(t)$ denote the total recruit inventory at time period t, and Y_s the number of units surviving the first s years of service. Let P denote the transition probability matrix, π_0 denote the recruitment vector, and $(1-\alpha)$ denote the failure rate. Then, the linear goal programming model (18)-(20) can be stated as follows:

$$Minimize \quad \sum_{t=1}^{T} (\, C^+(t)g^+(t) + C^-(t)g^-(t)\,) \tag{34}$$

$$s.t. \quad \sum_{s=1}^{t} Y_s R_{t+1-s} + g^-(t) - g^+(t) = G(t); \qquad t=1,\dots,T, \tag{35}$$

$$Y_s = \{ p^{s-1} + \alpha p^{s-2} + \dots + \alpha^{s-1} I \} \pi_0. \tag{36}$$

Notice the resemblance of (35) and (36) to (21) and (22).

Other constraints such as limits on the number of recruits for some or all of the periods under study can be added. The technique is applied to a case study based on the Irish Telephone System. Numerical results are presented in Young and Abodunde [1979].

Conclusions

Markov models are valuable tools in manpower planning. Stochastic and deterministic models are developed and successfully used to predict future manpower structures, to derive policies to achieve a specific manpower structure in the near and far future, and to provide insight to "what if" questions. Historical data are used to estimate the TPM. The estimated TPM is not very accurate, so the results from Markov models must be used cautiously. Deterministic models are preferred by some practitioners since they believe that forecasts are not accurate enough to justify the complexity of the stochastic models. Even though, Markov models are very helpful in manpower planning they do not take into account costs or conflicting objectives.

Manpower planning is closely linked to linear programming theory. Manpower planning problems can be represented as a linear goal-programs, where limitations and conflicting goals are formulated as constraints with objectives to minimize the deviations from these limitations. Several practical models have used this approach for solving manpower problems. This approach has proven to be very efficient in providing satisfactory results. In goal programming model, goal priorities have a significant role in determining how closely each goal is matched, and consequently, the solution profile. One of the suggested techniques for determining the objective weights, that reflect the priorities, is to systematically compare the main factors at different levels of the problem. Most of the data used in these goal programs rest on assumptions and predictions about several human and economic factors, e.g. officer survival rate. To properly incorporate these nondeterministic factors, Markov models are used to quantify these predictions to be used as goal-program constants.

The Prototype Model

The model described below is designed so that two reports can be generated and used for decision making. The inputs to the model are the following:

(i) the force structure,
(ii) inventory targets,
(iii) promotion targets, and
(iv) transition rules for personnel who move through the system.
The output tables take the following form:

Personnel Inventory Report						
Time	Grade	Policy Group	Target (Input)	Inventory Produced By The Model	Deviation	% Deviation

Personnel Promotion Report			
Time	Grade	Target % promoted at this (Time,Grade) combination (input)	% Promotions produced by the model

Graphical Display

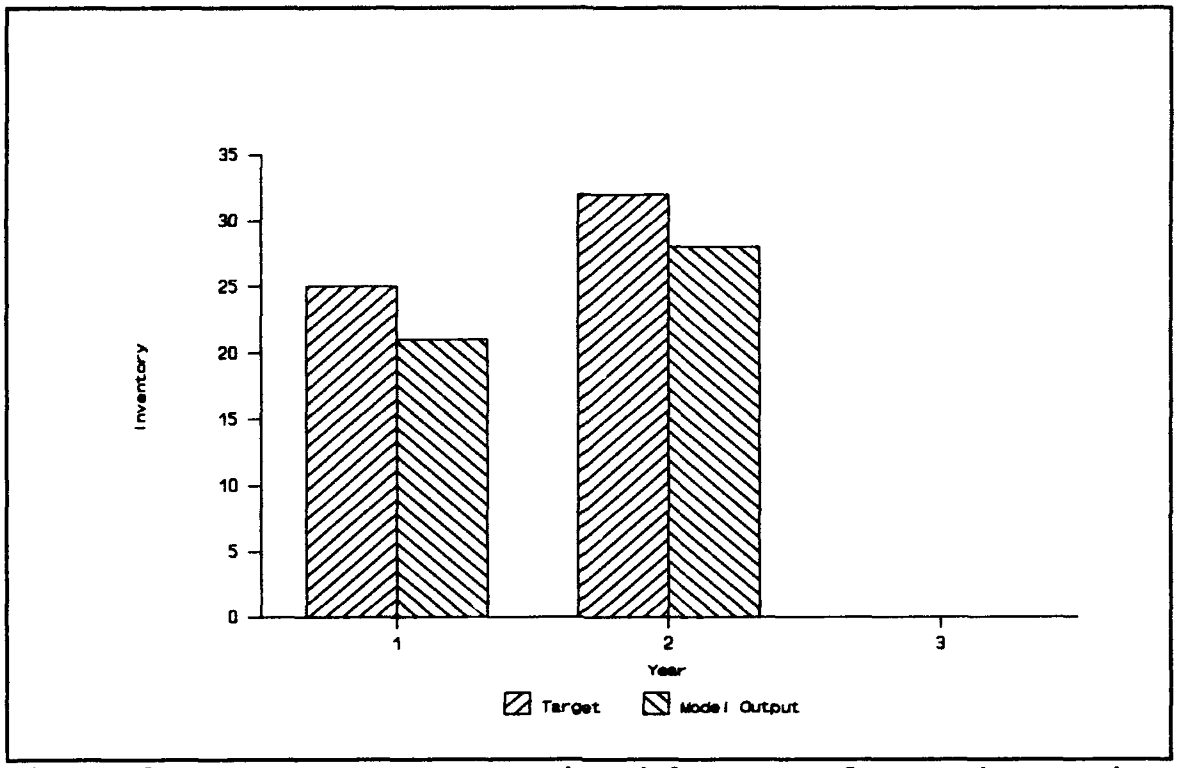
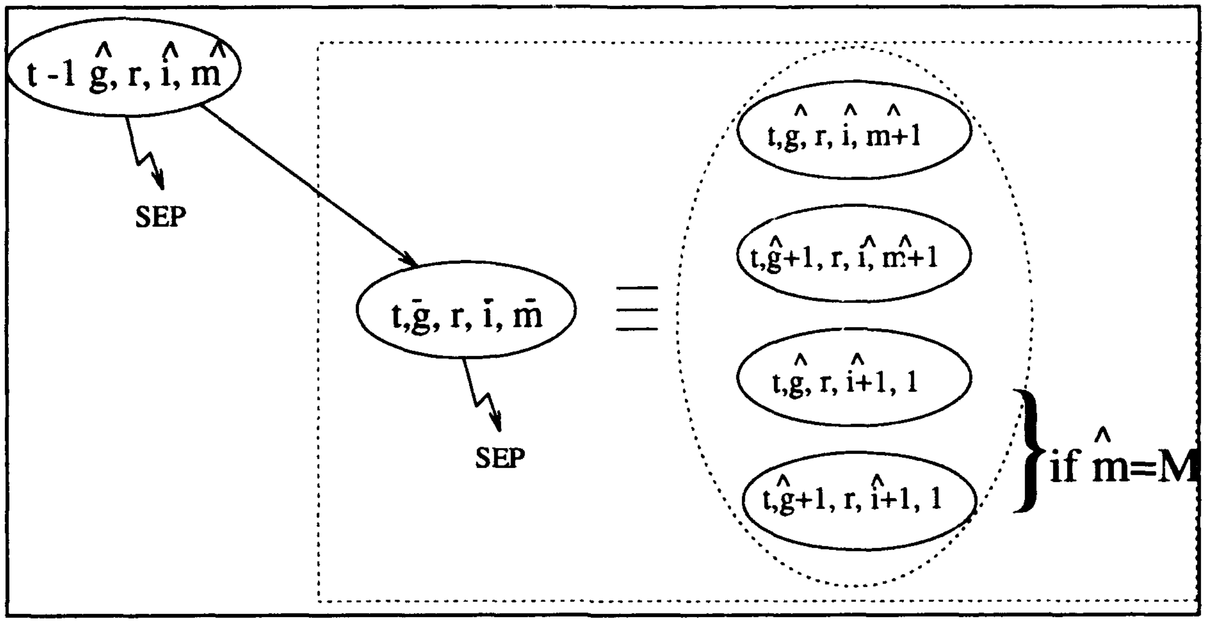


Figure 2. Inventory target and model output for grade g and policy group i

Subscripts

t- denotes the time period (years)
g- denotes the grade
r- denotes the location (r=1 conUS, r=2 not conUS)
i- denotes a policy group
m- denotes the number of years in policy group i
It is assumed that all personnel can be characterized by a 5-tuple given by the 5 subscripts. We will refer to a 5-tuple (t,g,r,i,m) as a state from which a transition can be made to another state.

Graphically this may be viewed as follows:



$t-1\ \hat{g}, r, \hat{i}, \hat{m}$
SEP
$t, \bar{g}, r, \bar{i}, \bar{m}$
SEP
$\equiv$
$t, \hat{g}, r, \hat{i}, \hat{m}+1$
$t, \hat{g}+1, r, \hat{i}, \hat{m}+1$
$t, \hat{g}, r, \hat{i}+1, 1$
$t, \hat{g}+1, r, \hat{i}+1, 1$
if $\hat{m}=M$

The rules for the possible transitions are input to the model.

Node Names

(t,g,r,i,m)	- state
ACC	- denotes the origin node for all accessions
SEP	- denotes a destination node for all separations
BAL	- denotes a balance node to balance the supply and

demand in the model. This is required for a true network model of the form AX=r where A is a node-arc incidence matrix.

Arc Types

$(t,g,r,i,m;t+1,\bar{g},\bar{r},\bar{i},\bar{m})$ - arcs which corresponding to moving from state (t,g,r,i,m) to state $(t+1,\bar{g},\bar{r},\bar{i},\bar{m})$
(ACC;t,1,r,1,1) arc from the node ACC to state (t,1,r,1,1).
(t,g,r,i,m;SEP) arc from the node (t,g,r,i,m) to the separation node.
(H,g,r,i,m;BAL) arc from state (H,g,r,i,m) to the node BAL.
(SEP;BAL) arc from node SEP to node BAL.

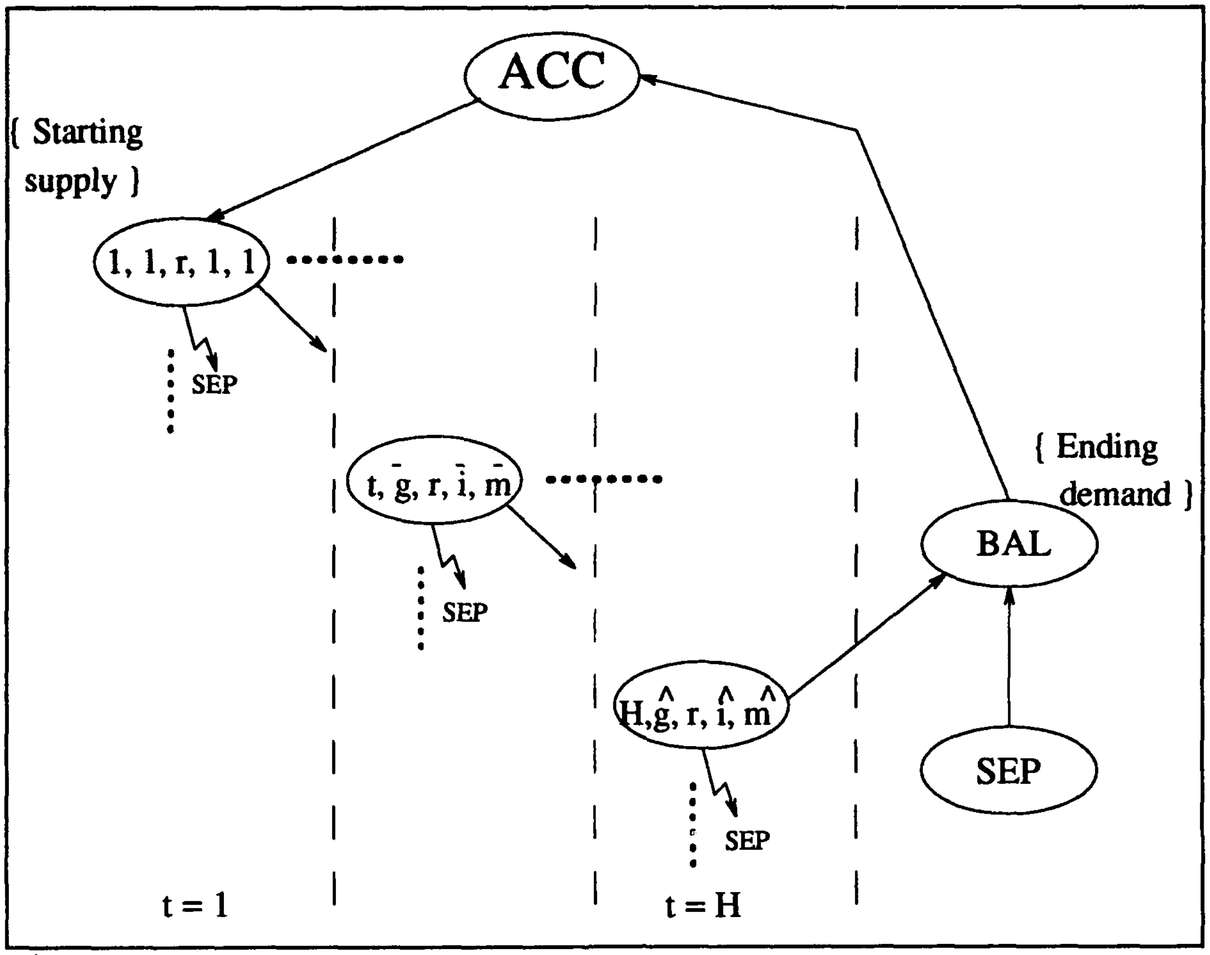


Figure 3. General structure

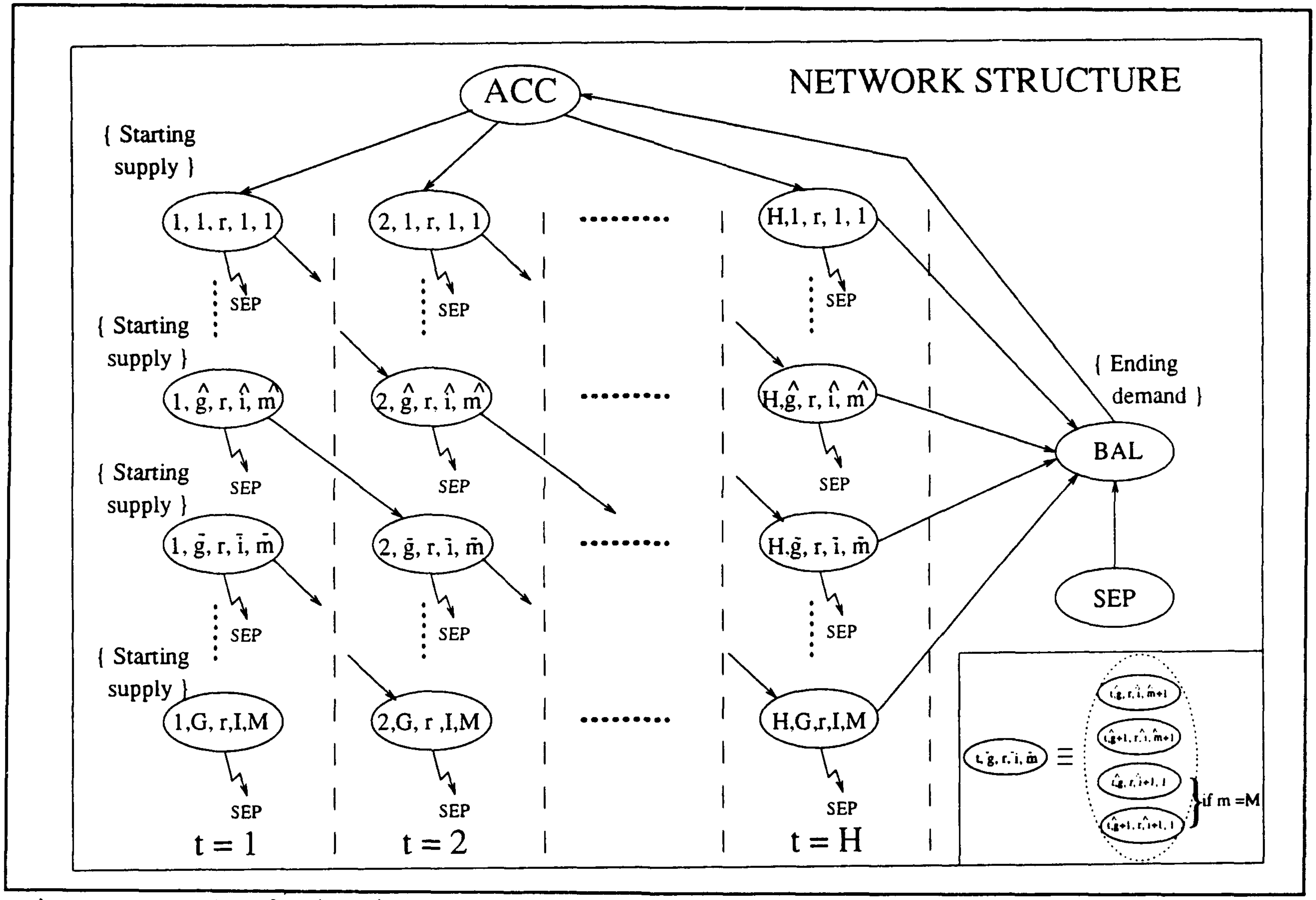


Figure 4. Network structure

Constants and Sets

H- planning horizon (t=1,...,H)
T(g,r,i,m) - the set of nodes $(\bar{g},\bar{r},\bar{i},\bar{m})$ for which there exists an arc $(t,g,r,i,m;t+1,\bar{g},\bar{r},\bar{i},\bar{m})$ for t=1,...,H-1. This is sometimes called the to set or the after set.
F(g,r,i,m) - the set of nodes $(t-1,\bar{g},\bar{r},\bar{i},\bar{m})$ for which there exists an arc $(t-1,\bar{g},\bar{r},\bar{i},\bar{m};t,g,r,i,m)$ for t=2,...,H. This is sometimes called the from set or the before set.
RHS(g,r,i,m) - number of people in the state (1,g,r,i,m)
ES(t,g) - force structure (people)
N(t,g,i) - inventory targets (people)
PROM(t,g) - Promotion Targets (%)
W_1, W_2, W_3, W_4 - weights for deviations from targets.

Decision Variables

$X(t,g,r,i,m;t+1,\bar{g},\bar{r},\bar{i},\bar{m})$ - number of people on the arc $(t,g,r,i,m;t+1,\bar{g},\bar{r},\bar{i},\bar{m})$
I(t,g,i) - number of people who leave state (t,g,i)
P(t,g) - number of people who leave state (t,g) and are promoted
A(t,r,1,1) - number of people on arc (ACC,t,1,r,1,1)
B(g,r,i,m) - number of people on arc (H,g,r,i,m;BAL)
C - number of people on arc (BAL,ACC)
D - number of people on arc (SEP,BAL)
S(t,g,r,i,m) - number of people on arc (t,g,i,m;SEP)
IO(t,g,i) - number of people over the desired inventory level
IU(t,g,i) - number of people under the desired inventory level
PO(t,g) - number of people over the desired promotion % goal
PU(t,g) - number of people under the desired promotion % goal

Constraints

Conservation of Flow (t,g,i,m)=(1,1,1,1)

$$\sum_{(\bar{g},\bar{r},\bar{i},\bar{m})\in T(2,r,i,m)} X(1,1,r,1,1;2,\bar{g},\bar{r},\bar{i},\bar{m}) + S(1,1,r,1,1) - A(1,r,1,1) = RHS(1,r,1,1) \quad \forall \quad (r,i,m)$$

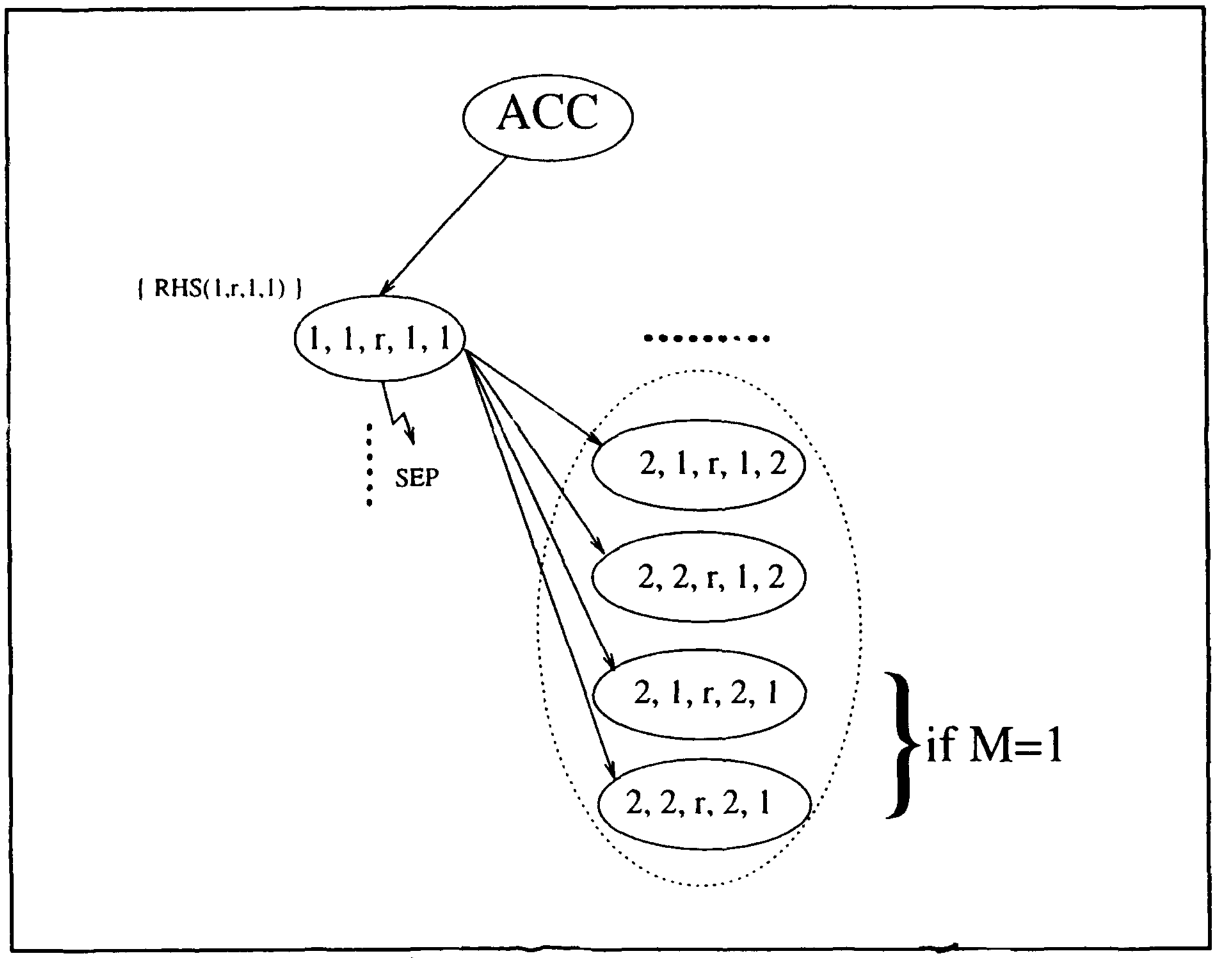


Figure 5. Conservation of flow t=1, g=1

Conservation of Flow t=1, $(g,i,m) \neq (1,1,1)$

$$\sum_{(\overline{g},\overline{r},\overline{i},\overline{m}) \in T(g,r,i,m)} X(1,g,r,i,m;2,\overline{g},\overline{r},\overline{i},\overline{m}) + S(1,g,r,i,m)$$
$$= RHS(g,r,i,m) \quad \forall \quad (g>1,r,i,m)$$

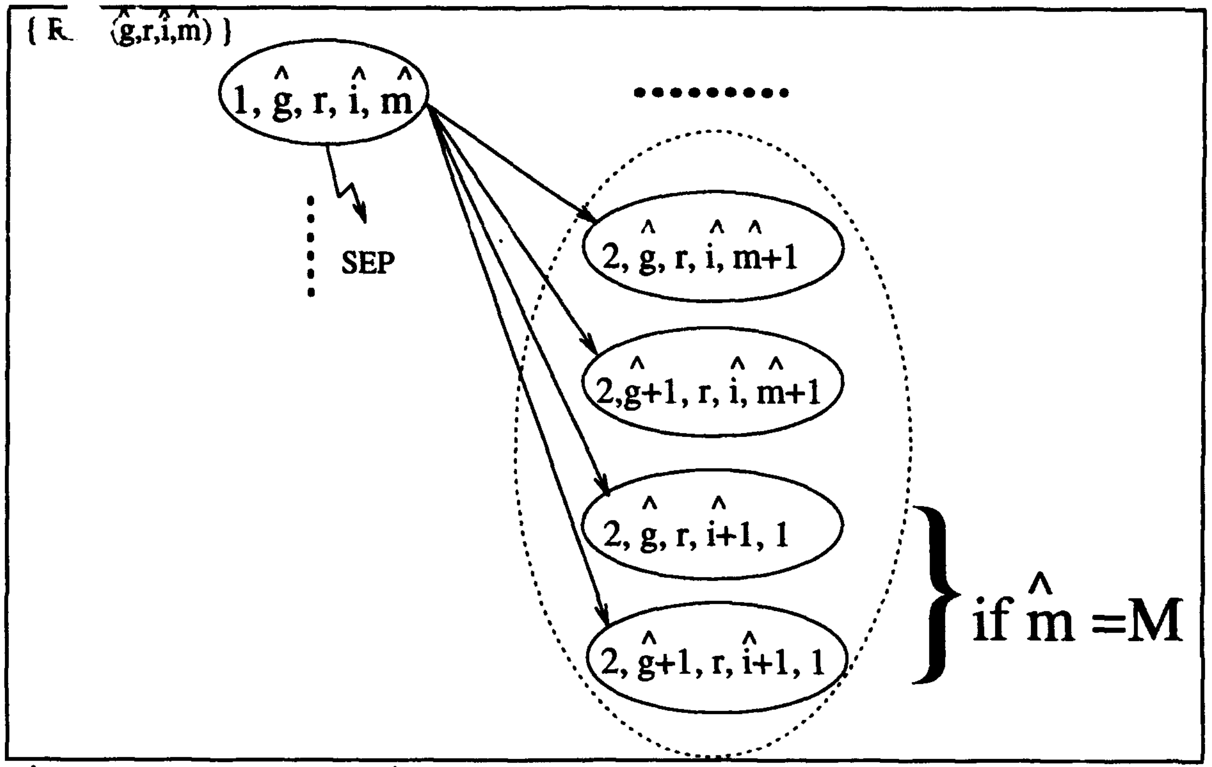


Figure 6. Conservation of flow t=1, g>1

Conservation of Flow $1<t<H$, $g=1$

$$\sum_{(\bar{g},\bar{r},\bar{i},\bar{m})\in T(1,r,i,m)} X(t,1,r,i,m;t+1,\bar{g},\bar{r},\bar{i},\bar{m}) + S(t,1,r,i,m)$$
$$- \sum_{(g,r,i,m)\in F(1,r,i,m)} X(t-1,1,\bar{r},\bar{i},\bar{m};t,1,r,i,m)$$
$$- A(t,r,i,m) = 0 \quad \forall \quad (1<t<H,1,r,i,m)$$

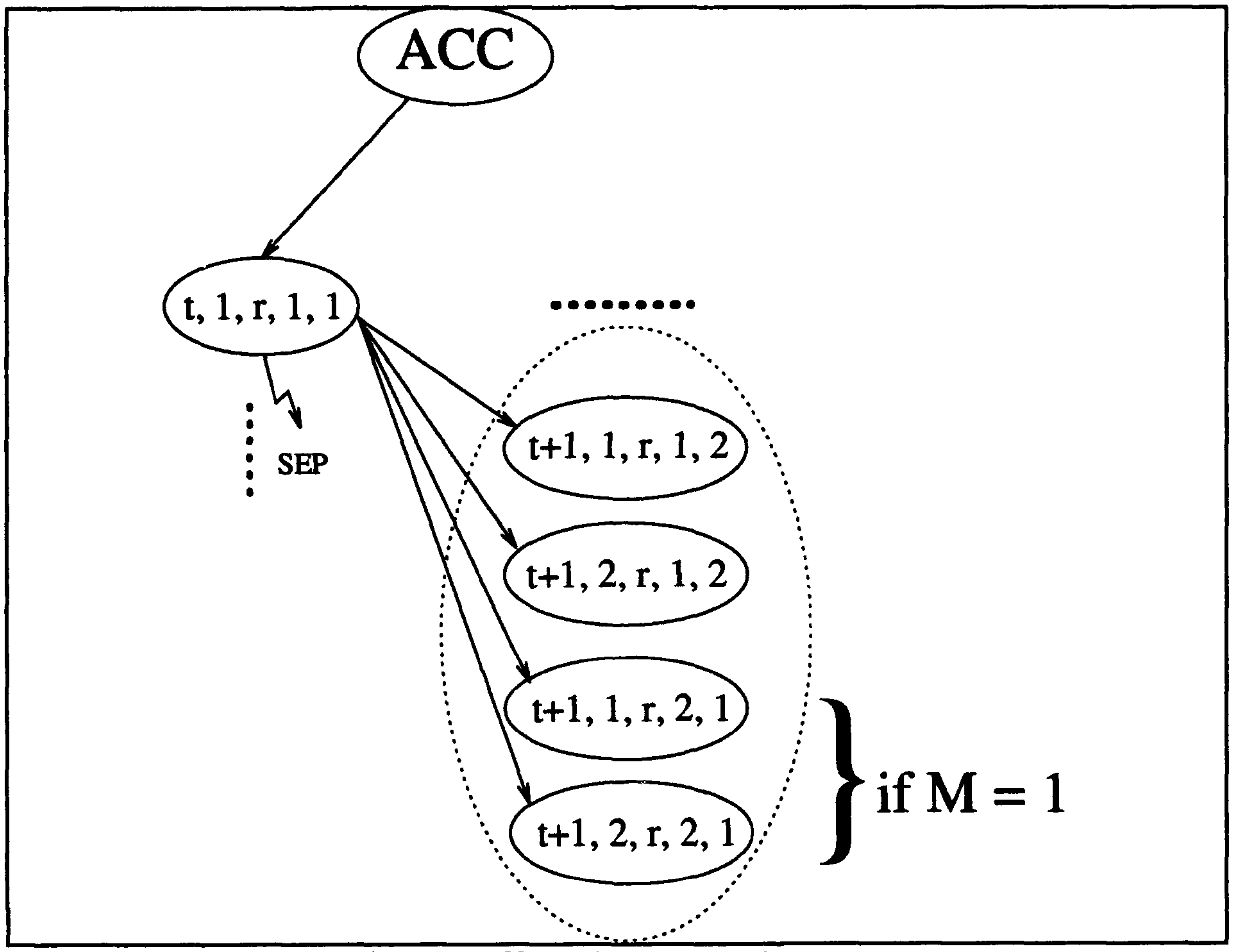


Figure 7. Conservation of flow 1<t<H, g=1

Conservation of Flow $1<t<H$, $(g,i,m)\neq(1,1,1)$

$$\sum_{(\overline{g},\overline{r},\overline{i},\overline{m})\in T(g,r,i,m)} X(t,g,r,i,m;t+1,\overline{g},\overline{r},\overline{i},\overline{m}) + S(t,1,r,i,m)$$
$$- \sum_{(\overline{g},\overline{r},\overline{i},\overline{m})\in F(1,r,i,m)} X(t-1,\overline{g},\overline{r},\overline{i},\overline{m};t,g,r,i,m) = 0 \quad \forall \quad (1<t<H,g,r,i,m)$$

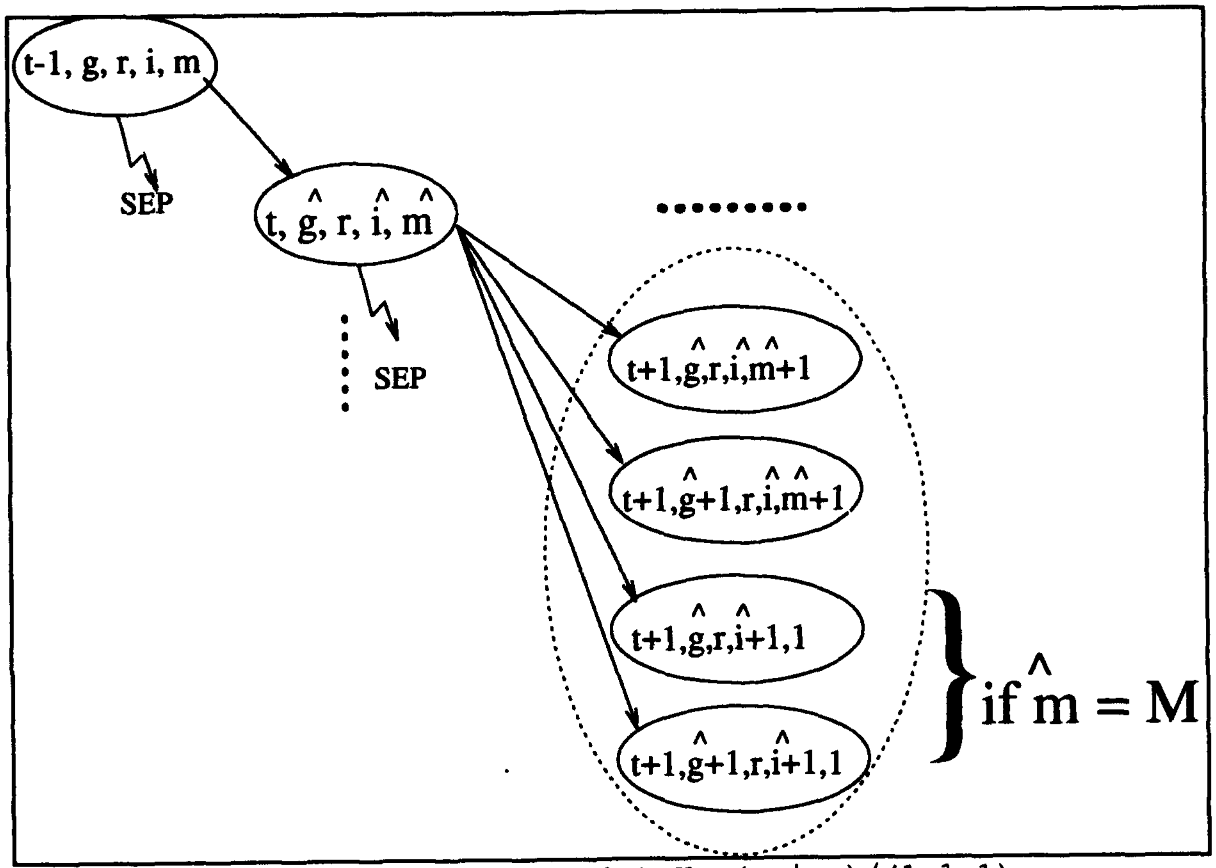


Figure 8. Conservation of flow $1<t<H$, $(g,i,m)\neq(1,1,1)$

Conservation of Flow t=H, (g,i,m)=(1,1,1)

$$S(H,1,r,1,1)+B(1,r,1,1) - A(H,r,1,1) = 0$$

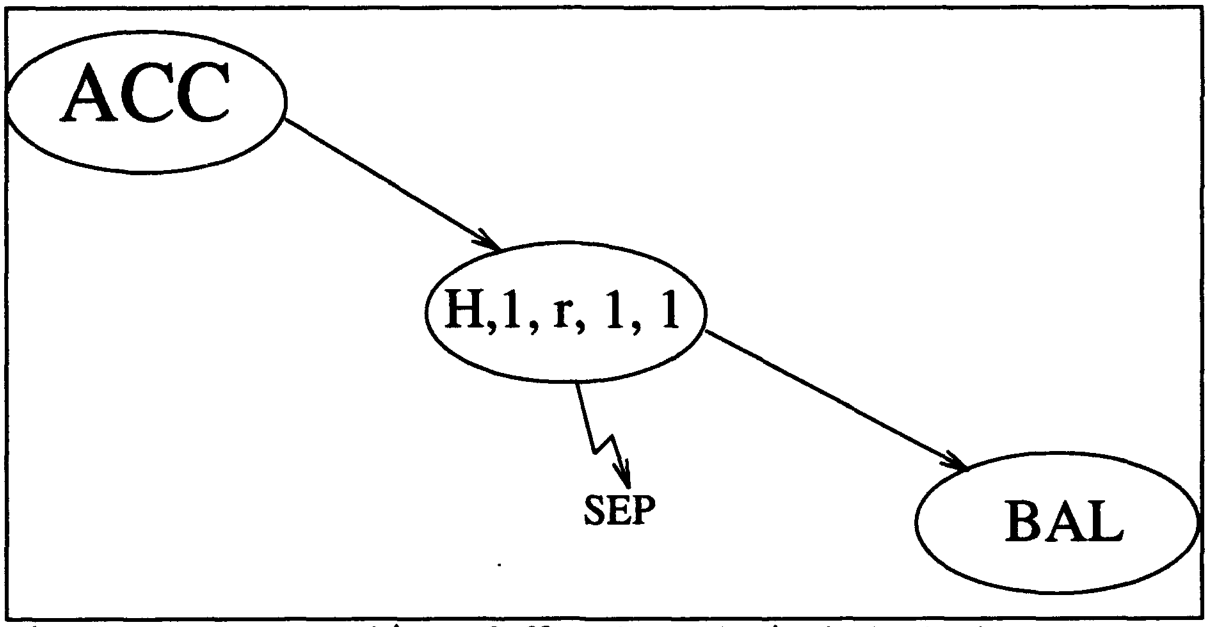


Figure 9. Conservation of flow t=H, (g,i,m)=(1,1,1)

Conservation of Flow t=H, (g,i,m)≠(1,1,1)

$$S(H,g,r,i,m) - \sum_{(\bar{g},\bar{r},\bar{i},\bar{m})\in F(g,r,i,m)} X(H-1,\bar{g},\bar{r},\bar{i},\bar{m};H,g,r,i,m) + B(g,r,i,m) = 0, \quad \forall \ (g,r,i,m)$$

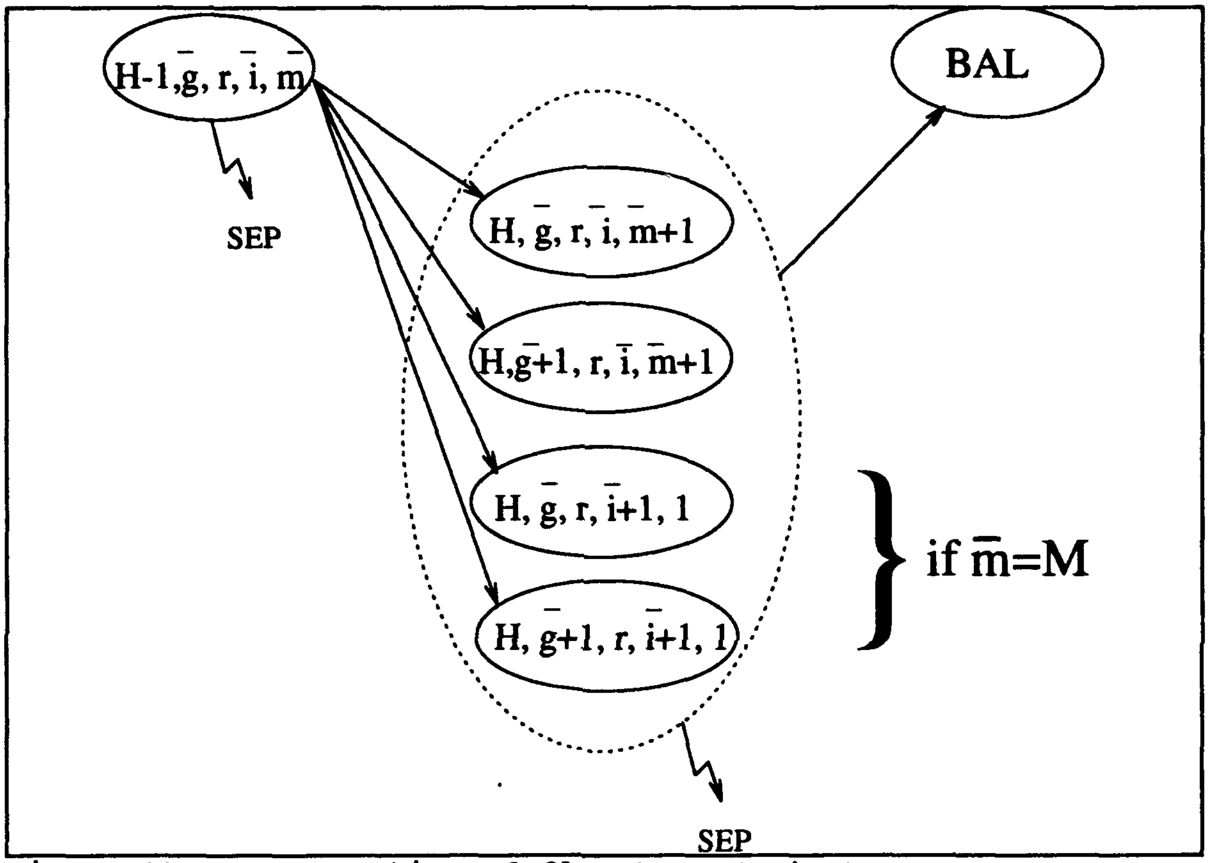

H-1, $\bar{g}$, r, $\bar{i}$, $\bar{m}$
BAL
SEP
H, $\bar{g}$, r, $\bar{i}$, $\bar{m}$+1
H, $\bar{g}$+1, r, $\bar{i}$, $\bar{m}$+1
H, $\bar{g}$, r, $\bar{i}$+1, 1
H, $\bar{g}$+1, r, $\bar{i}$+1, 1
if $\bar{m}$=M
SEP

Figure 10. Conservation of flow t=H, (g,i,m)≠(1,1,1)

Conservation of Flow at ACC.

$$\sum_{(t,r,i,m)} A(t,r,i,m) - C = 0$$

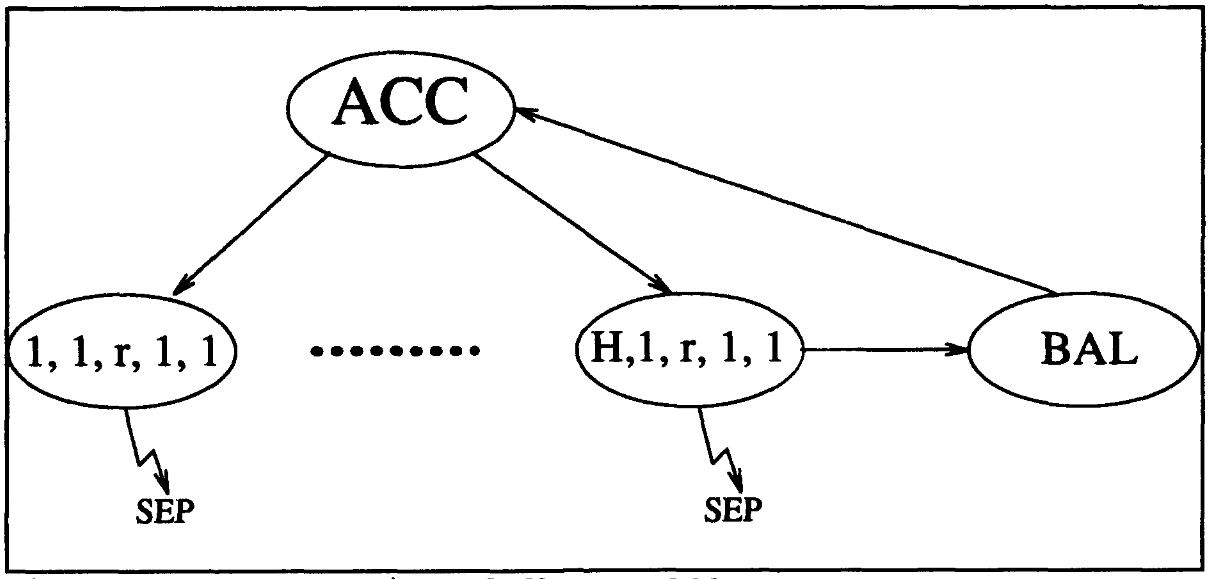


Figure 11. Conservation of flow at ACC

Conservation of Flow at SEP

$$D - \sum_{t,g,r,i,m)} S(t,g,r,i,m) = 0$$

Inventory Constraints t<H

$$\sum_{r,m} \sum_{(\overline{g},\overline{r},\overline{i},\overline{m})\in T(g,r,i,m)} X(t,g,r,i,m;t+1,\overline{g},\overline{r},\overline{i},\overline{m})$$
$$+\sum_{r,m} S(t,g,r,i,m) - I(t,g,i) = 0, \quad \forall \quad (t<H,g,i)$$

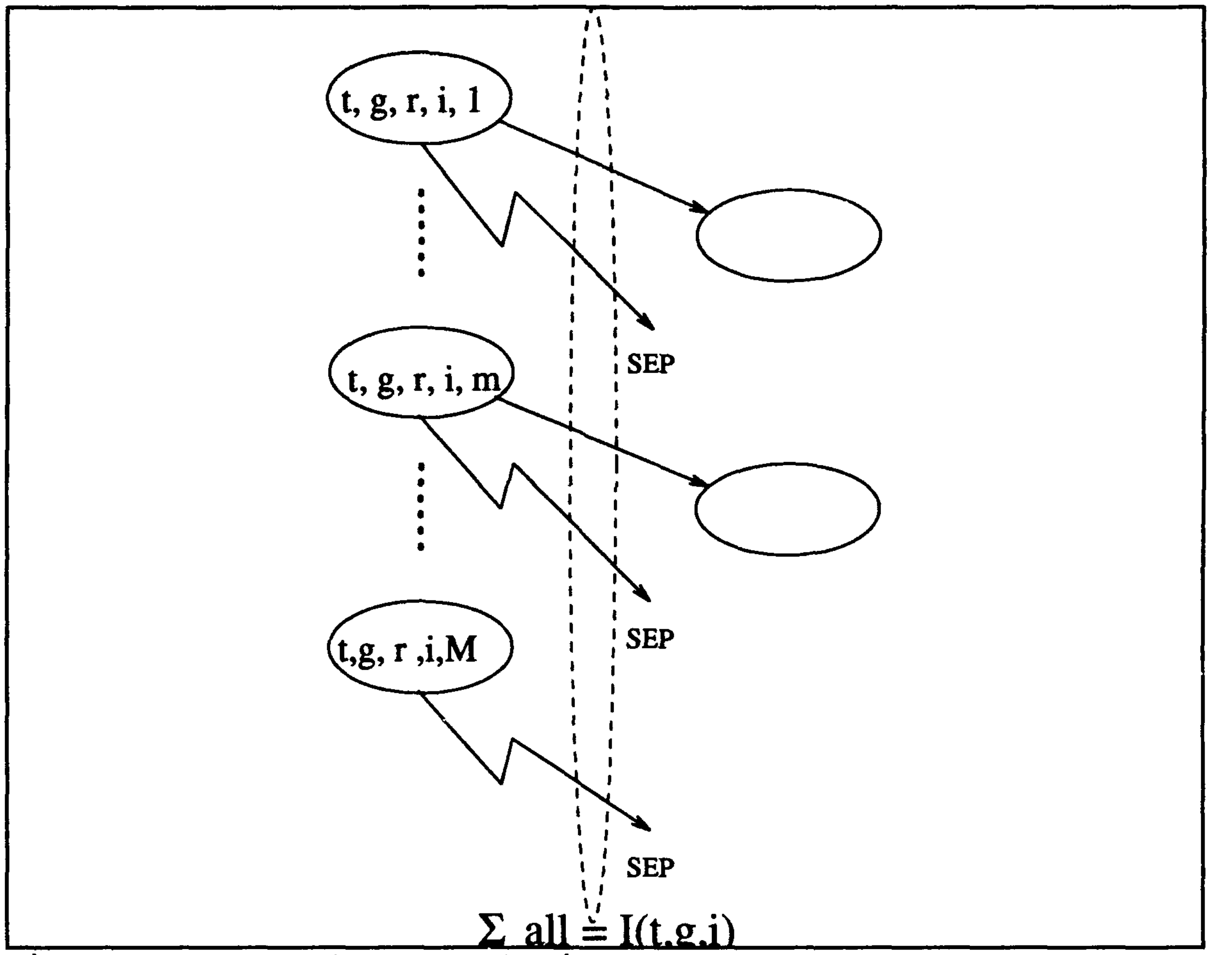


Figure 12. Inventory constraints t<H

Inventory Constraints t=H

$$\sum_{r,m} B(g,r,i,m) + \sum_{r,m} S(H,g,r,i,m) - I(H,g,i) = 0, \quad \forall \ (g,i)$$

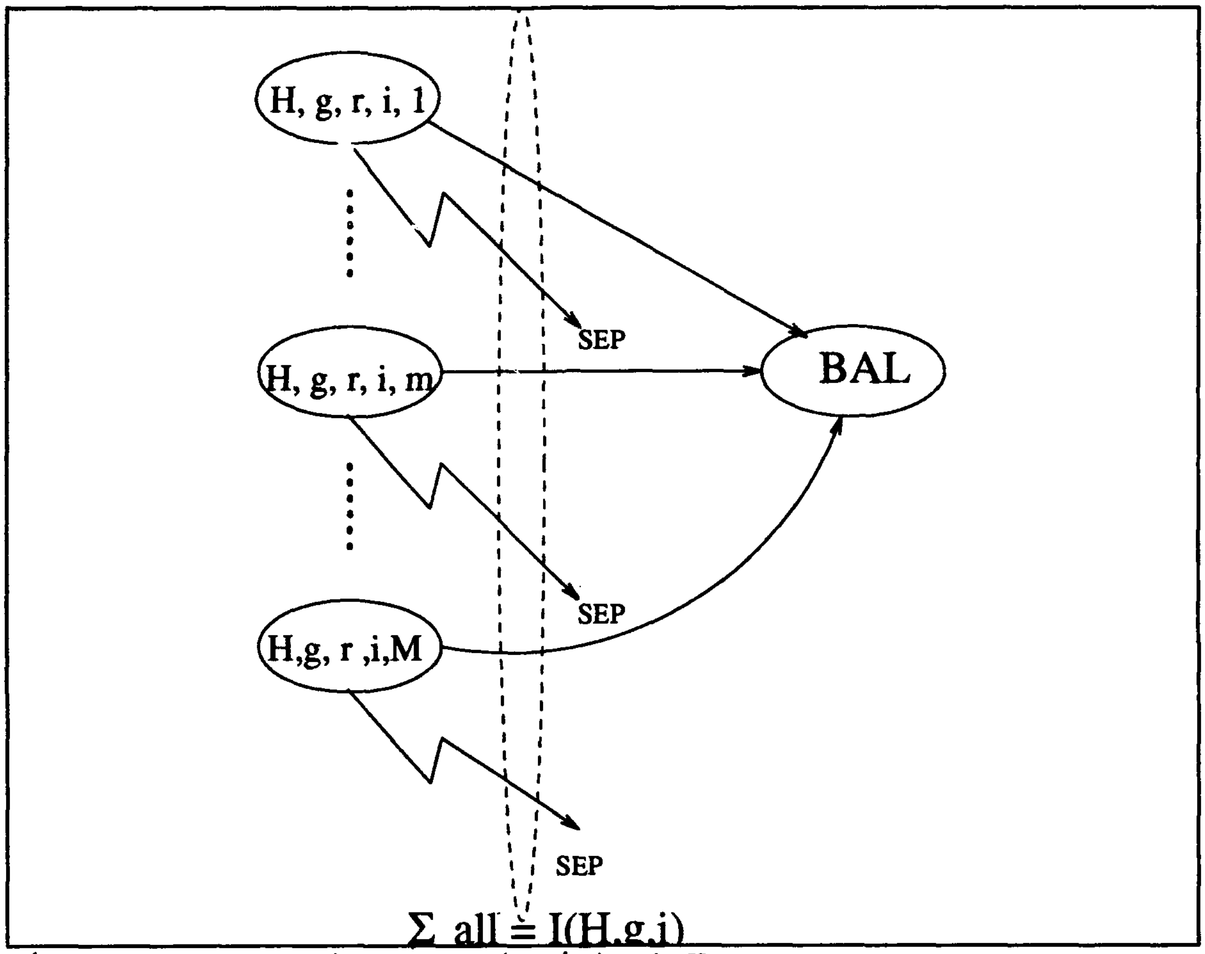


Figure 13. Inventory constraints t=H

Force Structure Constraints

$$\sum_{i} I(t,g,i) = ES(t,g), \quad \forall \quad (t,g)$$

Promotion Accounting Constraints t<H,g<G

$$\sum_{(r,i,m)} \sum_{(g+1,\overline{r},\overline{i},\overline{m}) \in (g,r,i,m)} X(t,g,r,i,m;t+1,g+1,\overline{r},\overline{i},\overline{m}) - P(t,g) = 0 \quad \forall \quad (t<H, g<G)$$

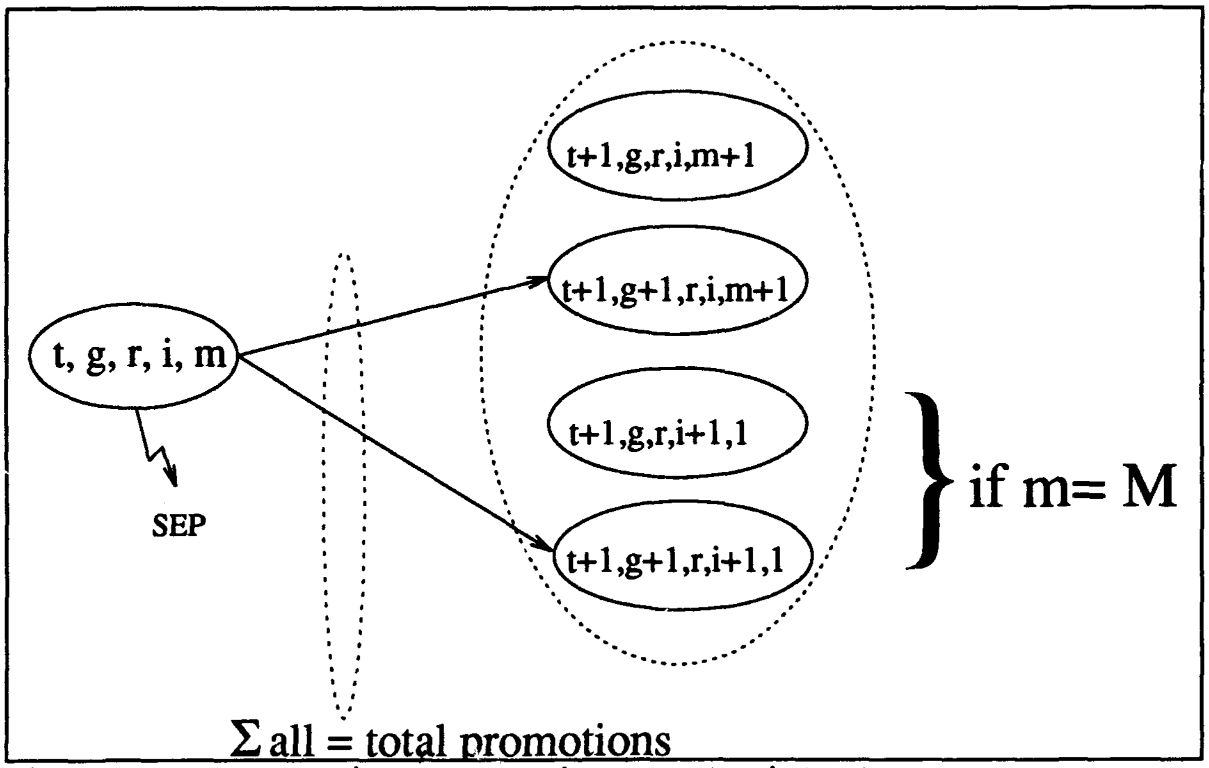


Figure 14. Promotion accounting constraints t<H, g<G

Deviation From Inventory Goal

$$I(t,g,i) + IU(t,g,i) - IO(t,g,i) = N(t,g,i), \quad \forall \ (t,g,i)$$

Deviation From Promotion Goal

$$P(t,g) + PU(t,g) - PO(t,g) = PROM(t,g)ES(t,g), \quad \forall \ (t<H,g)$$

Objective Function

$$\text{minimize} \sum_{(t,g,i)} (W_1 IU(t,g,i) + W_2 IO(t,g,i)) + \sum_{(t,g)} (W_3 PU(t,g) + W_4 PO(t,g))$$

Constraint Layout

	Name	Size
States t,g,r,i,m	ST	H.G.R.I.M
ACC	ACC	1
SEP	SEP	1
BAL	BAL	1
Inventory	INV	H.G.I
Force	FOR	H.G
Promotion	PRO	(H-1).(G-1)
Inventory Goal	IG	H.G.I
Promotion Goal	PG	(H-1).(G-1)

Total Constraints = H.G.R.I.M + 2.H.G.I + 2.(H-1).(G-1)+ H.G + 3

Column Layout

Row Name	Column Name	Special Condition
	$X(t,g,r,i,m;t+1,\bar{g},\bar{r},\bar{i},\bar{m})$	$t<H$
ST(t,g,r,i,m)	1	
$ST(t+1,\bar{g},\bar{r},\bar{i},\bar{m})$	-1	
INV(t,g,i)	1	
PRO(t,g)	1	if$\bar{g}>g$

Row Name	Column Name	Special Condition
	I(t,g,i)	
INV(t,g,i)	-1	
FOR(t,g)	1	
IG(t,g,i)	1	

Row Name	Column Name	Special Condition
	P(t,g)	$t<H$
PRO(t,g)	-1	
PG(t,g)	1	

Row Name	Column Name	Special Condition
	A(t,r,i,m)	
ACC	1	
ST(t,1,r,i,m)	-1	

Row Name	Column Name	Special Condition
	B(g,r,i,m)	
BAL	-1	
ST(H,g,r,i,m)	1	
INV(H,g,i)	1	

Row Name	Column Name	Special Condition
	C	
BAL	1	
ACC	-1	

Row Name	Column Name	Special Condition
	D	
SEP	1	
BAL	-1	

Row Name	Column Name	Special Condition
	S(t,g,r,i,m)	
SEP	-1	
ST(t,g,r,i,m)	1	
INV(t,g,i)	1	

Row Name	Column Name	Special Condition
	IO(t,g,i)	
IG(t,g,i)	-1	
OBJ	W2	

Row Name	Column Name	Special Condition
	IU(t,g,i)	
IG(t,g,i)	1	
OBJ	W1	

Row Name	Column Name	Special Condition
	PO(t,g)	
PG(t,g)	-1	
OBJ	W4	

Row Name	Column Name	Special Condition
	PU(t,g)	
PG(t,g,i)	1	
OBJ	W3	

References

Abodunde, T., and S. McClean, 1980, "Production Planning for a Manpower System with a Constant Level of Recruitment," Applied Statistics, Vol. 29, No.1, 43-49.

Bartholomew, D., 1982, Stochastic Models for Social Processes, John Wiley and Sons, New York, New York.

Bres, E., D. Burns, A. Charnes, and W. Cooper, 1980, "A Goal Programming Model For Planning Officer Accessions," Management Science, Vol. 26, No. 8, 773-783.

Collins, R., S. Gass and E. Rosendahl, 1983, "The ASCAR Model for Evaluating Military Manpower Policy," Interfaces, Vol. 13, No. 3, 44-53.

Edwards J., 1983, "A Survey of Manpower Planning Models and Their Applications," Journal of Operational Research Society, Vol. 37, No. 11, 1031-1040.

Gass, S., 1986, "A Process for Determining priorities and Weights for Large-Scale Linear Goal Programmers," Journal of Operational Research, Vol. 37, No. 8, 779-785.

Hall, R. W., 1986, "Graphical Models for Manpower Planning," International Journal of Production Research, Vol. 24, No. 5, 1267-1282.

Holz, B. and J. Worth, 1980, "Improving Strength Forecasts: Support For Army Manpower Management," Interfaces, Vol. 10, No. 6, 37-52.

Smith, A., and Bartholomew, D., 1988, "Manpower Planning in the United Kingdom: An Historical Review," Journal of Operational Research Society, Vol. 39, No. 3, 235-284.

Raghavendra, B., 1991, "A Bivariate Model for Markov Manpower Planning Systems," Journal of Operational Research Society, Vol. 42, No. 7, 565-570.

Vassiliou P., 1976, "A Markov Chain Model for Wastage in Manpower Systems," Operational Research Quarterly, Vol. 27, No. 1, 57-70.

Vajda, S., 1976, Mathematics of manpower Planning, John Wiley New York.

Young, A. and T. Abodunde, 1979, "Personnel Recruitment Policies and Long-term Production Planning," Journal of Operational Research Society, Vol. 30, No. 3, 225-236.

Zanakis S. and M. Maret, 1980, "A Markov Chain Application to Manpower Supply Planning," Journal of Operational Research Society, Vol. 31, No. 12, 1095-1102.

Zanakis S. and M. Maret, 1981, "A Markovian Goal Programming Approach to Aggregate Manpower Planning," Journal of Operational Research Society, Vol. 32, No. 1, 55-63.

Appendix A
Model Generator and Test Case

93/01/24
12:49:44

mp.for

```
$NOTRUNCATE
*  MP developes a manpower planning model
*
*  Written by J. Kennington
*             December 1992
*             (214)-768-3278
*
*  Input Files:
*
*  1.  H G R I M  (free format)
*
*      where H - denotes the planning horizon
*            G - denotes the number of grades
*            R = 2  (r = 1 => conus, r = 2, otherwise)
*            I - denotes the number of policy groups
*            M - denotes the maximum number of years in a policy group
*
*  2.  g r i m RHS(g,r,i,m)  (1/line free format)
*
*  3.  t g ES(t,g)           (1/line free format)
*
*  4.  t g i N(t,g,i)        (1/line free format)
*
*  5.  t g PROM(t,g)         (1/line free format in %)
*
*  7.  W1 W2 W3 W4           (1/line free format)
*
*  Output Files:
*
*  8.  MPS File
*
*  9.  Output File
*
*  Temporary Storage
*
*  10. Temp File
*
       open( 1,file='1.dat',status='old')
       open( 2,file='2.dat',status='old')
       open( 3,file='3.dat',status='old')
       open( 4,file='4.dat',status='old')
       open( 5,file='5.dat',status='old')
       open( 7,file='7.dat',status='old')
       open( 8,file='8.mps')
       open( 9,file='out')
       open(10,file='temp')

       call input

       call name

       call rows

       call column

       call rhss

       call end

       stop 'normal exit'

       end
*
*
```

```
*
      subroutine input
*  input reads the input files
*
$INCLUDE:'mp.inc'

      character*1 char
      integer     t,g,r,i,m,l
      real        vrhs, ves, vnn, vprom
      integer     kgrim, ktgi, ktg

 1001 format(a1)
*
*  File 1
*
      read (1,1001) char
      read (1,*)     HH,GG,RR,II,MM
      if( (HH.lt.1).or.(HH.gt.zzzH) ) call error(1)
      if( (GG.lt.1).or.(GG.gt.zzzG) ) call error(1)
      if( (RR.lt.1).or.(RR.gt.zzzR) ) call error(1)
      if( (II.lt.1).or.(II.gt.zzzI) ) call error(1)
      if( (MM.lt.1).or.(MM.gt.zzzM) ) call error(1)
      im   = II*MM
      rim  = RR*im
      grim = GG*rim
      hg   = HH*GG
      hgi  = hg*II
      gi   = GG*II
*
*  File 2
*
      read (2,1001) char
      do 10 l=1,grim
      RHS(l) = 0.
   10 continue
*     do 20 while(grim.ne.0)
   11 continue
        read (2,*,end=30) g,r,i,m,vrhs
        if( (g.lt. 1) .or. (r.lt. 1) .or. (i.lt. 1) .or. (m.lt. 1) )
     $  call error(2)
        if( (g.gt.GG) .or. (r.gt.RR) .or. (i.gt.II) .or. (m.gt.MM) )
     $ call error(2)
        if(vrhs.lt.0.0) call error(2)
        l       = kgrim(g,r,i,m)
        RHS(l) = vrhs
   20 continue
      go to 11
   30 continue
*
*  File 3
*
      read (3,1001) char
      do 40 l=1,hg
        ES(l) = 0.
   40 continue
*     do 50 while(hg.ne.0)
   41 continue
        read (3,*,end=60) t,g,ves
        if( (t.lt. 1) .or. (g.lt. 1) ) call error(3)
        if( (t.gt.HH) .or. (g.gt.GG) ) call error(3)
        if(ves.lt.0.) call error(3)
        l       = ktg(t,g)
        ES(l) = ves
   50 continue
```

```
      go to 41
   60 continue
*
*  File 4
*
      read (4,1001) char
      do 70 l=1,hgi
        NN(l) = 0.
   70 continue
*     do 80 while(hgi.ne.0)
   71 continue
        read (4,*,end=90) t,g,i,vnn
        if( (t.lt. 1) .or. (g.lt. 1) .or. (i.lt. 1) ) call error(4)
        if( (t.gt.HH) .or. (g.gt.GG) .or. (i.gt.II) ) call error(4)
        if(vnn.lt.0.) call error(4)
        l      = ktgi(t,g,i)
        NN(l) = vnn
   80 continue
      go to 71
   90 continue
*
*  File 5
*
      read (5,1001) char
      do 100 l=1,hg
        PROM(l) = 0.
  100 continue
*     do 110 while(hg.ne.0)
  101 continue
        read (5,*,end=120) t,g,vprom
        if( (t.lt. 1) .or. (g.lt. 1) ) call error(5)
        if( (t.gt.HH) .or. (g.gt.GG) ) call error(5)
        if( (vprom.lt.0.) .or. (vprom.gt.100.) ) call error(5)
        l = ktg(t,g)
        PROM(l) = vprom
  110 continue
      go to 101
  120 continue
*
*  File 7
*
      read (7,1001) char
      read (7,*,end=999) W1, W2, W3, W4
      return
  999 continue
      call error(7)
      end
*
*
*
      subroutine name
*  name writes the name card
*
      write(8,1001)
 1001 format('NAME',t15,'MANPOWER')
      return
      end
*
*
*
      subroutine rows
*  rows writes the ROWS section of the MPS file
*
      write(8,1001)
```

```
 1001 format('ROWS')

      call OBJ
      call ST
      call ACC
      call SEP
      call BAL
      call INV
      call FOR
      call PRO
      call IG
      call PG

      return
      end
*
*
*
      subroutine column
*  column generates the COLUMNS section of the MPS file
*
      write(8,1001)
 1001 format('COLUMNS')

      call varX
      call varI
      call varP
      call varA
      call varB
      call varC
      call varD
      call varS
      call varIO
      call varIU
      call varPO
      call varPU

      return
      end
*
*
*
      subroutine rhss
*  rhss writes the RHS section of the MPS file
*
      write(8,1001)
 1001 format('RHS')
      call ST1
      call BAL1
      call FOR1
      call IG1
      call PG1
      return
      end
*
*
*
      subroutine end
*  end writes the ENDATA line in the MPS file
*
      write(8,1001)
 1001 format('ENDATA')
      return
      end
```

```
*
*
*
      subroutine OBJ
*  OBJ generates the row OBJ
*
      write(8,1001)
 1001 format(t2,'N',t5,'OBJ')
      return
      end
*
*
*
      subroutine ST
*  ST generates the ST(t,g,r,i,m) rows
*
$INCLUDE:'mp.inc'

      character*1 name(8)
      integer     t,g,r,i,m,l
      integer     ktgrim

      call blank(name)

      do 50 t=1,HH
        do 40 g=1,GG
          do 30 r=1,RR
            do 20 i=1,II
              do 10 m=1,MM
                l = ktgrim(t,g,r,i,m)
                call rowST(l,name)
                call rowpnt(name)
   10         continue
   20       continue
   30     continue
   40   continue
   50 continue

      return
      end
*
*
*
      subroutine ACC
*  ACC generates the row ACC
*
      implicit logical (a-z)
      character*1 name(8)

      call blank(name)

      name(1) = 'A'
      name(2) = 'C'
      name(3) = 'C'

      call rowpnt(name)

      return
      end
*
*
*
      subroutine SEP
*  SEP generates the row SEP
```

```
*
      implicit logical (a-z)
      character*1 name(8)

      call blank(name)

      name(1) = 'S'
      name(2) = 'E'
      name(3) = 'P'

      call rowpnt(name)

      return
      end
*
*
*
      subroutine BAL
*  BAL generates the row BAL
*
      implicit logical (a-z)
      character*1 name(8)

      call blank(name)

      name(1) = 'B'
      name(2) = 'A'
      name(3) = 'L'

      call rowpnt(name)

      return
      end
*
*
*
      subroutine INV
*  INV generates the INV(t,g,i) rows
*
$INCLUDE:'mp.inc'

      character*1 name(8)
      integer     t,g,i,l
      integer     ktgi

      call blank(name)

      do 30 t=1,HH
        do 20 g=1,GG
          do 10 i=1,II
            l = ktgi(t,g,i)
            call rowINV(l,name)
            call rowpnt(name)
   10     continue
   20   continue
   30 continue

      return
      end
*
*
*
      subroutine FOR
*  FOR generates the FOR(t,g) rows
```

93/01/24
12:49:44

```
*
$INCLUDE:'mp.inc'

      character*1 name(8)
      integer     t,g,l
      integer     ktg

      call blank(name)

      do 20 t=1,HH
        do 10 g=1,GG
          l = ktg(t,g)
          call rowFOR(l,name)
          if(t.eq.1) then
            call rowpnt1(name)
          else
            call rowpnt(name)
          end if
   10   continue
   20 continue

      return
      end
*
*
*
      subroutine PRO
*  PRO generates the PRO(t,g) rows
*
$INCLUDE:'mp.inc'

      character*1 name(8)
      integer     t,g,l
      integer     ktg

      call blank(name)

      do 20 t=1,HH-1
        do 10 g=1,GG-1
          l = ktg(t,g)
          call rowPRO(l,name)
          call rowpnt(name)
   10   continue
   20 continue

      return
      end
*
*
*
      subroutine IG
*  IG  generates the IG(t,g,i) rows
*
$INCLUDE:'mp.inc'

      character*1 name(8)
      integer     t,g,i,l
      integer     ktgi

      call blank(name)

      do 30 t=1,HH
        do 20 g=1,GG
          do 10 i=1,II
```

mp.for

```
            l = ktgi(t,g,i)
            call rowIG(l,name)
            call rowpnt(name)
   10     continue
   20   continue
   30 continue

      return
      end
*
*
*
      subroutine PG
*  PG  generates the PG(t,g) rows
*
$INCLUDE:'mp.inc'

      character*1 name(8)
      integer     t,g,l
      integer     ktg

      call blank(name)

      do 20 t=1,HH-1
        do 10 g=1,GG-1
          l = ktg(t,g)
          call rowPG(l,name)
          call rowpnt(name)
   10   continue
   20 continue

      return
      end
*
*
*
      subroutine varX
*  varX generates the X columns
*
$INCLUDE:'mp.inc'

      character*1 xname(8),rname1(8),rname2(8),rname3(8),rname4(8)
      integer     t,g,r,i,m,l1,l2,l3,l4
      integer     ktg,ktgi,ktgrim

      call blank(xname)
      call blank(rname1)
      call blank(rname2)
      call blank(rname3)
      call blank(rname4)

      do 50 t = 1,HH-1
        do 40 g = 1,GG
          l4 = ktg(t,g)
          call rowPRO(l4,rname4)
          do 30 r = 1,RR
            do 20 i = 1,II
              l3 = ktgi(t,g,i)
              call rowINV(l3,rname3)
              do 10 m = 1,MM
                l1 = ktgrim(t,g,r,i,m)
                call rowST(l1,rname1)
*
*  Rules For Arcs
```

93/01/24
12:49:44

mp.for

```
*
            if(m.lt.MM) then
*             status quo
              12 = ktgrim(t+1,g,r,i,m+1)
              call colX(11,12,xname)
              call rowST(12,rname2)
              call colum2(xname,rname1,1.,rname2,-1.)
              call colum1(xname,rname3,1.)
            end if
            if( (m.lt.MM) .and. (g.lt.GG) ) then
*             promotion
              12 = ktgrim(t+1,g+1,r,i,m+1)
              call colX(11,12,xname)
              call rowST(12,rname2)
              call colum2(xname,rname1,1.,rname2,-1.)
              call colum2(xname,rname3,1.,rname4,+1.)
            end if
            if( (m.eq.MM) .and. (i.lt.II) ) then
*             new policy group, same r
              12 = ktgrim(t+1,g,r,i+1,1)
              call colX(11,12,xname)
              call rowST(12,rname2)
              call colum2(xname,rname1,1.,rname2,-1.)
              call colum1(xname,rname3,1.)
*             new policy group, new r
              12 = ktgrim(t+1,g,3-r,i+1,1)
              call colX(11,12,xname)
              call rowST(12,rname2)
              call colum2(xname,rname1,1.,rname2,-1.)
              call colum1(xname,rname3,1.)
            end if
            if( (m.eq.MM) .and. (i.lt.II) .and. (g.lt.GG) ) then
*             promotion, new policy group, same r
              12 = ktgrim(t+1,g+1,r,i+1,1)
              call colX(11,12,xname)
              call rowST(12,rname2)
              call colum2(xname,rname1,1.,rname2,-1.)
              call colum2(xname,rname3,1.,rname4,+1.)
*             promotion, new policy, new group
              12 = ktgrim(t+1,g+1,3-r,i+1,1)
              call colX(11,12,xname)
              call rowST(12,rname2)
              call colum2(xname,rname1,1.,rname2,-1.)
              call colum2(xname,rname3,1.,rname4,+1.)
            end if
   10       continue
   20     continue
   30   continue
   40   continue
   50 continue
      return
      end
*
*
*
      subroutine varI
*  varI generates the I(t,g,i) columns
*
$INCLUDE:'mp.inc'

      character*1 xname(8), rname1(8),  rname2(8), rname3(8)
      integer     t,g,i,11,12
      integer     ktg, ktgi

      call blank(xname)
      call blank(rname1)
      call blank(rname2)
      call blank(rname3)

      do 30 t = 1,HH
        do 20 g = 1,GG
          12 = ktg(t,g)
          call rowFOR(12,rname2)
          do 10 i = 1,II
            11 = ktgi(t,g,i)
            call colI  (11,xname)
            call rowINV(11,rname1)
            call rowIG (11,rname3)
            call colum2(xname,rname1,-1.,rname2,1.)
            call colum1(xname,rname3,1.)
   10     continue
   20   continue
   30 continue
      return
      end
*
*
*
      subroutine varP
*  varP generates the P(t,g) columns
*
$INCLUDE:'mp.inc'

      character*1 xname(8),rname1(8),rname2(8)
      integer     t,g,1
      integer     ktg

      call blank(xname)
      call blank(rname1)
      call blank(rname2)

      do 20 t = 1,HH-1
        do 10 g = 1,GG-1
          1 = ktg(t,g)
          call colP(1,xname)
          call rowPRO(1,rname1)
          call rowPG (1,rname2)
          call colum2(xname,rname1,-1.,rname2,1.)
   10   continue
   20 continue
      return
      end
*
*
*
      subroutine varA
*  varA generates the A(t,r,i,m) columns
*
$INCLUDE:'mp.inc'

      character*1 xname(8),rname1(8),rname2(8)
      integer     t,r,i,m,1
      integer     ktrim,ktgrim

      call blank(xname)
      call blank(rname1)
      call blank(rname2)
      rname1(1) = 'A'
      rname1(2) = 'C'
      rname1(3) = 'C'

      do 40 t = 1,HH
        do 30 r = 1,RR
          do 20 i = 1,II
            do 10 m = 1,MM
              l = ktrim(t,r,i,m)
              call colA(l,xname)
              l = ktgrim(t,l,r,i,m)
              call rowST(l,rname2)
              call colum2(xname,rname1,1.,rname2,-1.)
 10         continue
 20       continue
 30     continue
 40   continue
      return
      end
*
*
*
      subroutine varB
*  varB generates the columns B(g,r,i,m)
*
$INCLUDE:'mp.inc'

      character*1 xname(8),rname1(8),rname2(8),rname3(8)
      integer     g,r,i,m,l1,l2,l3
      integer     ktgi, ktgrim, kgrim

      call blank(xname)
      call blank(rname1)
      call blank(rname2)
      call blank(rname3)

      rname2(1) = 'B'
      rname2(2) = 'A'
      rname2(3) = 'L'

      do 40 g = 1,GG
        do 30 r = 1,RR
          do 20 i = 1,II
            l3 = ktgi(HH,g,i)
            call rowINV(l3,rname3)
            do 10 m = 1,MM
              l1 = kgrim(g,r,i,m)
              call colB(l1,xname)
              l2 = ktgrim(HH,g,r,i,m)
              call rowST(l2,rname1)
              call colum2(xname,rname1,1.,rname2,-1.)
              call colum1(xname,rname3,1.)
 10         continue
 20       continue
 30     continue
 40   continue
      return
      end
*
*
*
      subroutine varC
*  varC generates the column C(BAL;ACC)
*
      implicit logical(a-z)
      character*1 xname(8), rname1(8), rname2(8)

      call blank(xname)
      call blank(rname1)
      call blank(rname2)

      xname(1) = 'C'

      rname1(1) = 'B'
      rname1(2) = 'A'
      rname1(3) = 'L'

      rname2(1) = 'A'
      rname2(2) = 'C'
      rname2(3) = 'C'

      call colum2(xname, rname1, 1.,rname2, -1.)
      return
      end
*
*
*
      subroutine varD
*  varD generates the column D(SEP;BAL)
*
      implicit logical(a-z)
      character*1 xname(8), rname1(8), rname2(8)

      call blank(xname)
      call blank(rname1)
      call blank(rname2)

      xname(1) = 'D'

      rname1(1) = 'S'
      rname1(2) = 'E'
      rname1(3) = 'P'

      rname2(1) = 'B'
      rname2(2) = 'A'
      rname2(3) = 'L'

      call colum2(xname, rname1, 1.,rname2, -1.)
      return
      end
*
*
*
      subroutine varS
*  varS generates the columns S(t,g,r,i,m)
*
$INCLUDE:'mp.inc'

      character*1 xname(8), rname1(8), rname2(8), rname3(8)
      integer     t,g,r,i,m,l1,l3
      integer     ktgi,ktgrim

      call blank(xname)
      call blank(rname1)
      call blank(rname2)
      call blank(rname3)

      rname2(1) = 'S'
      rname2(2) = 'E'
      rname2(3) = 'P'

      do 50 t = 1,HH
        do 40 g = 1,GG
          do 30 r = 1,RR
            do 20 i = 1,II
              13 = ktgi(t,g,i)
              call rowINV(13,rname3)
                do 10 m = 1,MM
                  11 = ktgrim(t,g,r,i,m)
                  call colS(11,xname)
                  call rowST(11,rname1)
                  call colum2(xname,rname1,1.,rname2,-1.)
                  call colum1(xname,rname3,1.)
   10           continue
   20         continue
   30       continue
   40     continue
   50 continue
      return
      end
*
*
*
      subroutine varIO
*   varIO generates the columns IO(t,g,i)
*
$INCLUDE:'mp.inc'

      character*1 xname(8), rname1(8), rname2(8)
      integer     t,g,i,l
      integer     ktgi

      call blank(xname)
      call blank(rname1)
      call blank(rname2)

      rname2(1) = 'O'
      rname2(2) = 'B'
      rname2(3) = 'J'

      do 30 t = 1,HH
        do 20 g = 1,GG
          do 10 i = 1,II
            l = ktgi(t,g,i)
            call colIO(l,xname)
            call rowIG(l,rname1)
            call colum2(xname,rname1,-1.,rname2,W2)
   10     continue
   20   continue
   30 continue
      return
      end
*
*
*
      subroutine varIU
*   varIU generates the columns IU(t,g,i)
*
$INCLUDE:'mp.inc'

      character*1 xname(8), rname1(8), rname2(8)
      integer     t,g,i,l
```

```
      integer     ktgi

      call blank(xname)
      call blank(rname1)
      call blank(rname2)

      rname2(1) = 'O'
      rname2(2) = 'B'
      rname2(3) = 'J'

      do 30 t = 1,HH
        do 20 g = 1,GG
          do 10 i = 1,II
            l = ktgi(t,g,i)
            call colIU(l,xname)
            call rowIG(l,rname1)
            call colum2(xname,rname1,1.,rname2,W1)
   10     continue
   20   continue
   30 continue
      return
      end
*
*
*
      subroutine varPO
*   varPO generates the columns PO(t,g)
*
$INCLUDE:'mp.inc'

      character*1 xname(8), rname1(8), rname2(8)
      integer     t,g,l
      integer     ktg

      call blank(xname)
      call blank(rname1)
      call blank(rname2)

      rname2(1) = 'O'
      rname2(2) = 'B'
      rname2(3) = 'J'

      do 20 t = 1,HH-1
        do 10 g = 1,GG-1
          l = ktg(t,g)
          call colPO(l,xname)
          call rowPG(l,rname1)
          call colum2(xname,rname1,-1.,rname2,W4)
   10   continue
   20 continue
      return
      end
*
*
*
      subroutine varPU
*   varPU generates the columns PU(t,g)
*
$INCLUDE:'mp.inc'

      character*1 xname(8), rname1(8), rname2(8)
      integer     t,g,l
      integer     ktg
      call blank(xname)
      call blank(rname1)
      call blank(rname2)

      rname2(1) = 'O'
      rname2(2) = 'B'
      rname2(3) = 'J'

      do 20 t = 1,HH-1
        do 10 g = 1,GG-1
          l = ktg(t,g)
          call colPU(l,xname)
          call rowPG(l,rname1)
          call colum2(xname,rname1,1.,rname2,W3)
   10     continue
   20   continue
      return
      end
*
*
*
      subroutine BAL1
*  BAL1 generates tjhe RHS for BAL
*
$INCLUDE:'mp.inc'

      character*1 name(8), abe(8)
      real        rhsd

      call blank(name)
      call blank(abe)

      abe(1) = 'A'
      abe(2) = 'B'
      abe(3) = 'E'

      name(1) = 'B'
      name(2) = 'A'
      name(3) = 'L'

      rhsd = -supply

      call colum1(abe,name,rhsd)

      return
      end
*
*
*
      subroutine ST1
*  ST1 generates the RHS for ST(l,g,r,i,m)
*
$INCLUDE:'mp.inc'

      character*1 name(8), abe(8)
      integer     g,r,i,m,l
      integer     ktgrim

      call blank(name)
      call blank(abe)

      abe(1) = 'A'
      abe(2) = 'B'
      abe(3) = 'E'
```

```
      supply = 0.

      do 40 g = 1,GG
        do 30 r = 1,RR
          do 20 i = 1,II
            do 10 m = 1,MM
              l = ktgrim(l,g,r,i,m)
              call rowST(l,name)
              call colum1(abe,name,rhs(l))
              supply = supply + rhs(l)
   10       continue
   20     continue
   30   continue
   40 continue
      return
      end
*
*
*
      subroutine FOR1
*  FOR1 generates the RHS for FOR(t,g)
*
$INCLUDE:'mp.inc'

      character*1 name(8), abe(8)
      integer     t,g,l
      integer     ktg

      call blank(name)
      call blank(abe)

      abe(1) = 'A'
      abe(2) = 'B'
      abe(3) = 'E'

      do 20 t = 1,HH
        do 10 g = 1,GG
          l = ktg(t,g)
          call rowFOR(l,name)
          call colum1(abe,name,ES(l))
   10   continue
   20 continue
      return
      end
*
*
*
      subroutine IG1
*  IG1 generates the RHS for IG(t,g,i)
*
$INCLUDE:'mp.inc'

      character*1 name(8), abe(8)
      integer     t,g,i,l
      integer     ktgi

      call blank(name)
      call blank(abe)

      abe(1) = 'A'
      abe(2) = 'B'
      abe(3) = 'E'
      do 30 t = 1,HH
        do 20 g = 1,GG
          do 20 i = 1,II
            l = ktgi(t,g,i)
            call rowIG(l,name)
            call columl(abe,name,NN(l))
   10     continue
   20   continue
   30 continue
      return
      end
*
*
*
      subroutine PG1
*  PG1 generates the RHS for PG(t,g)
*
$INCLUDE:'mp.inc'

      character*1 name(8), abe(8)
      integer     t,g,l,k
      integer     ktg
      real        realk

      call blank(name)
      call blank(abe)

      abe(1) = 'A'
      abe(2) = 'B'
      abe(3) = 'E'

      do 20 t = 1,HH-1
        do 10 g = 1,GG-1
          l = ktg(t,g)
          call rowPG(l,name)
          k     = PROM(l)*ES(l)/100.
          realk = k
          call columl(abe,name,realk)
   10     continue
   20   continue
   30 continue
      return
      end
*
*
*
      subroutine colX(l,m,name)
*  colX builds the column name for variable X(l;m)
*
      implicit logical(a-z)
      character*1 name(8)
      integer l,m,i(4),j(4),k

      call decom(l,i)
      call decom(m,j)

      do 10 k=1,4
        call char(i(k),name(k))
        call char(j(k),name(k+4))
   10 continue
      return
      end
*
*
```

```
*
      subroutine colI(l,name)
*  colI builds the column name for variable I(l)
*
      implicit logical(a-z)
      character*1 name(8)
      integer l,i(4)

      call decom(l,i)
      call char4(i,name,1)
      name(1) = 'I'
      return
      end
*
*
*
      subroutine colP(l,name)
*  colP builds the column name P(l)
*
      implicit logical (a-z)
      character*1 name(8)
      integer     l,i(4)

      call decom(l,i)
      call char4(i,name,1)
      name(1) = 'P'
      return
      end
*
*
*
      subroutine colA(l,name)
*  colA builds the column name A(l)
*
      implicit logical (a-z)
      character*1 name(8)
      integer     l,i(4)

      call decom(l,i)
      call char4(i,name,1)
      name(1) = 'A'
      return
      end
*
*
*
      subroutine colB(l,name)
*  colB builds the column name B(g,r,i,m)
*
      implicit logical (a-z)
      character*1 name(8)
      integer     l,i(4)

      call decom(l,i)
      call char4(i,name,1)
      name(1) = 'B'
      return
      end
*
*
*
      subroutine colS(l,name)
*  colS builds the column name S(t,g,r,i,m)
*
```

```
      implicit logical (a-z)
      character*1 name(8)
      integer     l,i(4)

      call decom(l,i)
      call char4(i,name,1)
      name(1) = 'S'
      return
      end
*
*
*
      subroutine colIO(l,name)
*  colIO builds the column name IO(t,g,i)
*
      implicit logical (a-z)
      character*1 name(8)
      integer     l,i(4)

      call decom(l,i)
      call char4(i,name,2)
      name(1) = 'I'
      name(2) = 'O'
      return
      end
*
*
*
      subroutine colIU(l,name)
*  colIU builds the column name IU(t,g,i)
*
      implicit logical (a-z)
      character*1 name(8)
      integer     l,i(4)

      call decom(l,i)
      call char4(i,name,2)
      name(1) = 'I'
      name(2) = 'U'
      return
      end
*
*
*
      subroutine colPO(l,name)
*  colPO builds the column name PO(t,g)
*
      implicit logical (a-z)
      character*1 name(8)
      integer     l,i(4)

      call decom(l,i)
      call char4(i,name,2)
      name(1) = 'P'
      name(2) = 'O'
      return
      end
*
*
*
      subroutine colPU(l,name)
*  colPU builds the column name PU(t,g)
*
      implicit logical (a-z)
```

```
      character*1 name(8)
      integer     l,i(4)

      call decom(l,i)
      call char4(i,name,2)
      name(1) = 'P'
      name(2) = 'U'
      return
      end
*
*
*
      subroutine rowST(l,name)
*  rowST builds the row name for ST(t,g,r,i,m)
*
      implicit logical (a-z)
      character*1 name(8)
      integer l,i(4)

      call decom(l,i)
      call char4(i,name,2)

      name(1) = 'S'
      name(2) = 'T'
      return
      end
*
*
*
      subroutine rowINV(l,name)
*  rowINV builds the row name for INV(l)
*
      implicit logical (a-z)
      character*1 name(8)
      integer l,i(4)

      call decom(l,i)
      call char4(i,name,3)

      name(1) = 'I'
      name(2) = 'N'
      name(3) = 'V'
      return
      end
*
*
*
      subroutine rowFOR(l,name)
*  rowFOR builds the row name for FOR(l)
*
      implicit logical (a-z)
      character*1 name(8)
      integer l,i(4)

      call decom(l,i)
      call char4(i,name,3)

      name(1) = 'F'
      name(2) = 'O'
      name(3) = 'R'
      return
      end
*
*
```

```
*
      subroutine rowPRO(l,name)
*  rowPRO builds the row name for PRO(l)
*
      implicit logical (a-z)
      character*1 name(8)
      integer l,i(4)

      call decom(l,i)
      call char4(i,name,3)

      name(1) = 'P'
      name(2) = 'R'
      name(3) = 'O'
      return
      end
*
*
*
      subroutine rowIG(l,name)
*  rowIG builds the row name for IG(l)
*
      implicit logical (a-z)
      character*1 name(8)
      integer l,i(4)

      call decom(l,i)
      call char4(i,name,2)

      name(1) = 'I'
      name(2) = 'G'
      return
      end
*
*
*
      subroutine rowPG(l,name)
*  rowPG builds the row name for PG(l)
*
      implicit logical (a-z)
      character*1 name(8)
      integer l,i(4)

      call decom(l,i)
      call char4(i,name,2)

      name(1) = 'P'
      name(2) = 'G'
      return
      end
*
*
*
      subroutine colum1(col,row,val)
*  colum1 writes one row of column data with only 1 value
*
      implicit logical (a-z)
      character*1 col(8), row(8)
      real        val

      write(8,1001) col, row, val
 1001 format(t5,8a1,t15,8a1,t25,f12.2)
      return
      end
```

```
*
*
*
      subroutine colum2(col,row1,val1,row2,val2)
*  colum2 writes one row of column data with 2 values
*
      implicit logical (a-z)
      character*1 col(8), row1(8), row2(8)
      real        val1, val2

      write(8,1001) col, row1, val1, row2, val2
 1001 format(t5,8a1,t15,8a1,t25,f12.2,t40, 8a1,t50,f12.2)
      return
      end
*
*
*
      subroutine char(i,c)
*  char converts integer data into character data
*
      implicit logical (a-z)
      integer          i
      character*1      c

      rewind 10
      write(10,1003) i
      rewind 10
      read(10, 1004) c

 1003 format(i1)
 1004 format(a1)

      return
      end
*
*
*
      subroutine char4(array,name,offset)
*  char4 converts 4 integers in array and places them into 4 positions in
*  name beginning at offset + 1
*
      implicit logical (a-z)
      integer          array(4), offset, k
      character*1      name(8)

      do 10 k=1,4
        call char(array(k),name(k+offset))
   10 continue

      return
      end
*
*
*
      subroutine blank(name)
*  blank initializes name to all blanks
*
      implicit logical (a-z)
      character*1 name(8)
      integer i

      do 10 i=1,8
        name(i) = ' '
   10 continue
```

```
      return
      end
*
*
*
      subroutine decom(l,j)
*  decom decomposes a 4 digit integer into the 4 digits and places them in array j
*
      implicit logical (a-z)
      integer l,j(4)

      j(1) =  l/1000
      j(2) = (l-j(1)*1000)/100
      j(3) = (l-j(1)*1000-j(2)*100)/10
      j(4) =  l-j(1)*1000-j(2)*100-j(3)*10

      return
      end
*
*
*
      subroutine rowpnt(name)
*  rowpnt writes one line in the ROWS section for rows of type E
*
      implicit logical (a-z)
      character*1 name(8)

      write(8,1001) name
 1001 format(t2,'E',t5,8a1)

      return
      end
*
*
*
      subroutine rowpntl(name)
*  rowpntl writes one line in the ROWS section for rows of type N
*
      implicit logical (a-z)
      character*1 name(8)

      write(8,1001) name
 1001 format(t2,'N',t5,8a1)

      return
      end
*
*
*
      function ktgrim(t,g,r,i,m)
*  ktgrim calculates the subscripts
*
$INCLUDE:'mp.inc'
      integer t,g,r,i,m
      integer ktgrim
      ktgrim = (t-1)*grim + (g-1)*rim + (r-1)*im + (i-1)*MM + m
      return
      end
*
*
*
      function ktrim(t,r,i,m)
*  ktrim calculates the subscripts
*
$INCLUDE:'mp.inc'
      integer t,r,i,m
      integer ktrim
      ktrim = (t-1)*rim + (r-1)*im + (i-1)*MM + m
      return
      end
*
*
*
      function kgrim(g,r,i,m)
*  kgrim calculates the subscripts
*
$INCLUDE:'mp.inc'
      integer g,r,i,m
      integer kgrim
      kgrim = (g-1)*rim + (r-1)*im + (i-1)*MM + m
      return
      end
*
*
*
      function ktgi(t,g,i)
*  ktgi calculates the subscripts
*
$INCLUDE:'mp.inc'
      integer t,g,i
      integer ktgi
      ktgi = (t-1)*gi + (g-1)*II + i
      return
      end
*
*
*
      function ktg(t,g)
*  ktg calculates the subscripts
*
$INCLUDE:'mp.inc'
      integer t,g
      integer ktg
      ktg = (t-1)*GG + g
      return
      end
*
*
*
      subroutine error(key)
*  error prints error messages
*
      implicit logical (a-z)
      integer          key

      go to (1,2,3,4,5,6,7) , key

    1 continue
      write(9,*) 'Error in input file 1.dat'
      stop       'Error in input file 1.dat'

    2 continue
      write(9,*) 'Error in input file 2.dat'
      stop       'Error in input file 2.dat'

    3 continue
```

```
      write(9,*) 'Error in input file 3.dat'
      stop       'Error in input file 3.dat'

    4 continue
      write(9,*) 'Error in input file 4.dat'
      stop       'Error in input file 4.dat'

    5 continue
      write(9,*) 'Error in input file 5.dat'
      stop       'Error in input file 5.dat'

    6 continue
      stop       'Error 6'

    7 continue
      write(9,*) 'Error in input file 7.dat'
      stop       'Error in input file 7.dat'

      end
\032
```

93/01/24
12:26:40

mp.inc

```
implicit logical (a-z)
integer    zzzH  ,zzzG  ,zzzR  ,zzzI  ,zzzM
parameter (zzzH=6,zzzG=6,zzzR=2,zzzI=6,zzzM=6)

common /blk1/ HH,GG,RR,II,MM
integer       HH,GG,RR,II,MM

common /blk2/ grim, rim, im, hg, hgi, gi
integer       grim, rim, im, hg, hgi, gi

common /blk3/ RHS(zzzG*zzzR*zzzI*zzzM), supply
real          RHS                     , supply

common /blk4/ ES(zzzH*zzzG),NN(zzzH*zzzG*zzzI)
real          ES           ,NN

common /blk5/ PROM(zzzH*zzzG)
integer       PROM

common /blck6/ W1, W2, W3, W4
real           W1, W2, W3, W4
```